Imperial College London

RISS
Resilient Information Systems Security

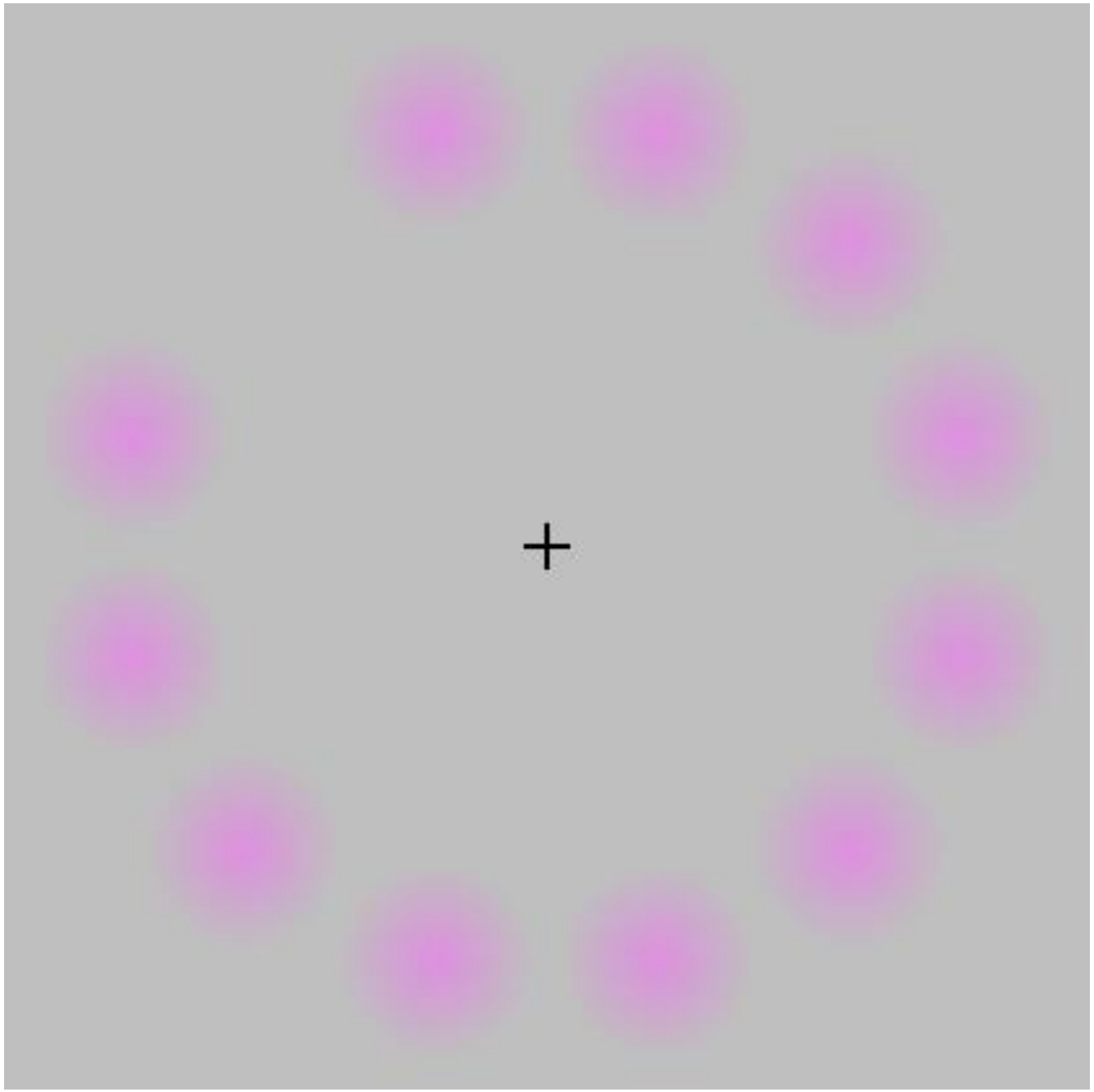
https://rissgroup.org

Adversarial Machine Learning

Luis Muñoz-González

l.munoz@imperial.ac.uk

20th December 2018



The Security of Machine Learning

Machine Learning systems can be compromised:

- Proliferation and sophistication of attacks and threats.
- Machine learning systems are one of the weakest parts in the security chain.
- Attackers can also use machine learning as a weapon.

Adversarial Machine Learning:

- Security of machine learning algorithms.
- Understanding the weaknesses of the algorithms.
- Proposing more resilient techniques.

Threats

Evasion Attacks:

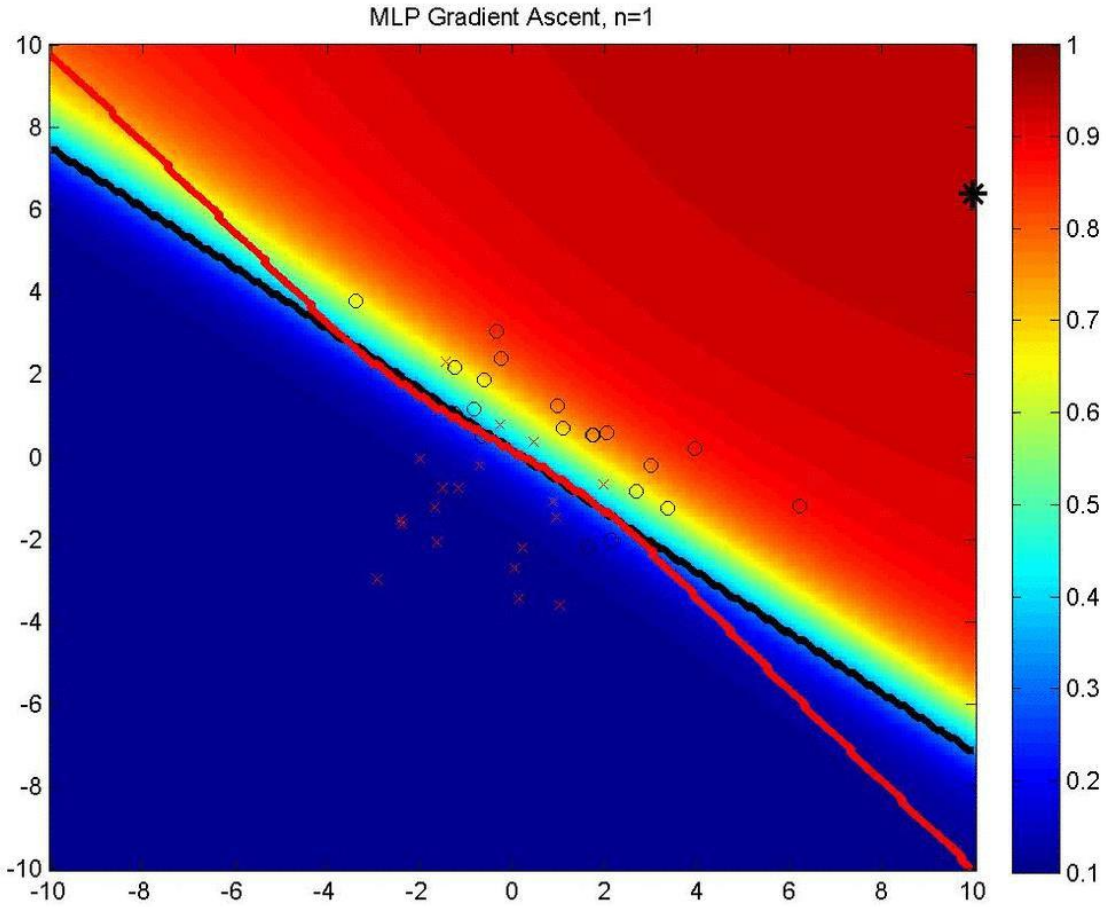
- Attacks at test time.
- The attacker aims to find the blind spots and weaknesses of the ML system to evade it.

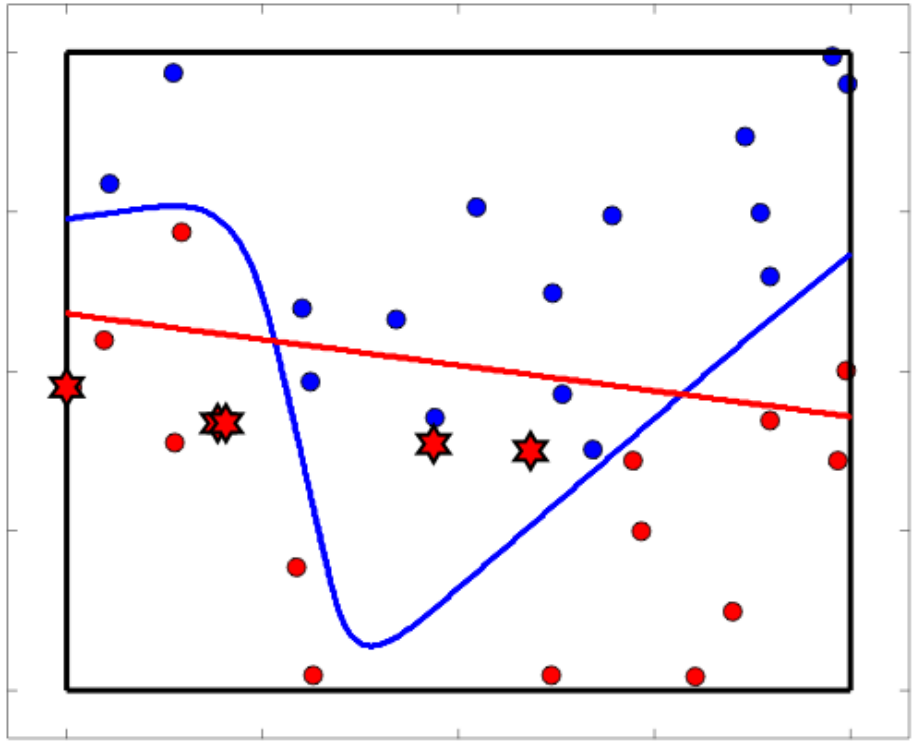
Poisoning Attacks:

- Compromise data collection.
- The attacker subverts the learning process.
- Degrades the performance of the system.
- Can facilitate future evasion.

Poisoning Attacks

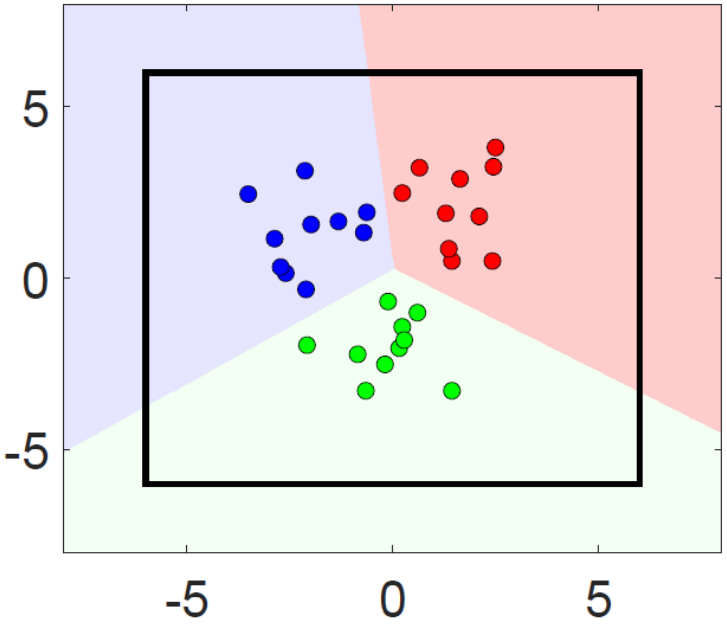
Poisoning Attacks



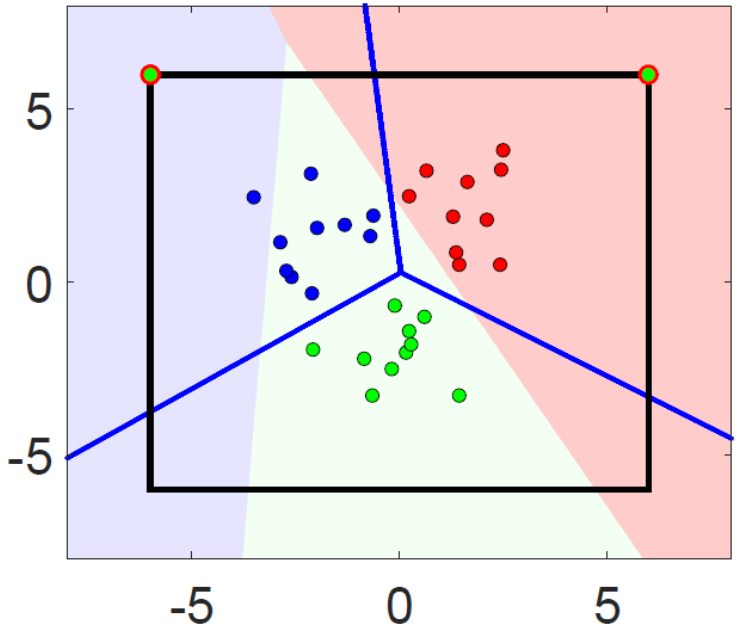


Types of Poisoning Attacks

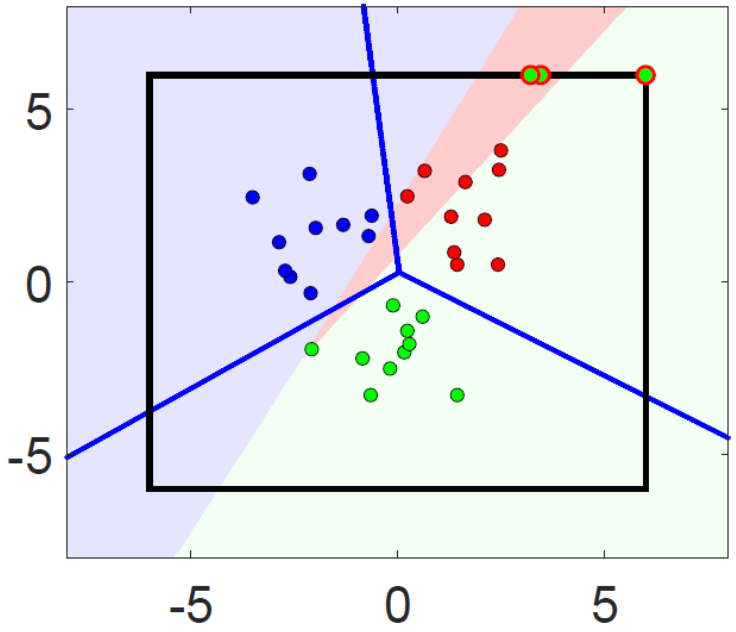
- **Error Generic:** aims to increase the overall classification error at test time.
- **Error Specific:** aims to produce some particular errors (misclassifications) at test time.



Clean dataset



Error Generic



Error Specific (attack red)

Optimal Poisoning Attacks

- The attacker aims to optimize some objective function (evaluated on a validation dataset) by introducing malicious examples in the training dataset used by the defender.
- The defender aims to learn the parameters of the model that optimise some objective function evaluated on the (poisoned) training dataset.
- The attacker's problem can be modelled as a **bi-level optimization problem** that can be efficiently solved with **back-gradient optimization**.

$$\mathcal{D}_p^* \in \arg\max_{\mathcal{D}_p} \; \mathcal{A}_{\text{val}}(\mathbf{w}^*, \mathcal{D}_{\text{val}}) \,,$$

$$\text{s.t.} \quad \mathbf{w}^* \in \arg\min_{\mathbf{w}} \; \mathcal{C}_{\text{tr}}(\mathbf{w}, \mathcal{D}_{\text{tr}} \cup \mathcal{D}_p)$$

Optimal Poisoning Attacks for Classification

$$\mathbf{x_p}^* \in \arg\max_{\mathbf{x_p} \in \mathcal{X}} \mathcal{C}_{\text{val}}(\mathbf{w}^*)\,,$$
$$\text{s.t. } \mathbf{w}^* \in \arg\min_{\mathbf{w}} \mathcal{C}_{\text{tr}}(\mathbf{w}, \mathcal{D}_{\text{tr}} \cup \{\mathbf{x_p}, y_p\})$$

- Biggio et al. "*Poisoning Attacks against Support Vector Machines*." ICML 2012.
- Mei and Zhu. "*Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners*." AAAI 2015.
- Xiao et al. "*Is Feature Selection Secure against Training Data Poisoning?*" ICML 2015.

- Poisoning points are learned following a **gradient ascent** strategy: $\nabla_{\mathbf{x_p}} \mathcal{C}_{\text{val}}(\mathbf{w}^*) = \left(\frac{\partial \mathbf{w}}{\partial \mathbf{x_p}}\right)^T \nabla_{\mathbf{w}} \mathcal{C}_{\text{val}}(\mathbf{w}^*)$
- Applying **Karush-Kuhn-Tucker** conditions $\nabla_{\mathbf{w}} \mathcal{C}_{\text{tr}}(\mathbf{w}, \mathbf{x_p}) = \mathbf{0}$ and **the implicit function theorem**:

$$\nabla_{\mathbf{x_p}} \mathcal{C}_{\text{val}} = -(\nabla_{\mathbf{x_p}} \nabla_{\mathbf{w}} \mathcal{C}_{\text{tr}})(\nabla^2_{\mathbf{w}} \mathcal{C}_{\text{tr}})^{-1} \nabla_{\mathbf{w}} \mathcal{C}_{\text{val}}$$

- Limited to a **restricted family of classifiers**.
- **Poor scalability** with the number of parameters of the model.

Optimal Poisoning Attacks for Classification

More efficient solution:

1) Don't invert matrices, use **conjugate gradient** instead:
   - More Stable.
   - Allows avoiding the computation of the Hessian.
2) **Divide and Conquer**:
   - Instead of computing $\nabla_{\mathbf{x_p}} \mathcal{C}_{\text{val}} = -(\nabla_{\mathbf{x_p}} \nabla_{\mathbf{w}} \mathcal{C}_{\text{tr}})(\nabla^2_{\mathbf{w}} \mathcal{C}_{\text{tr}})^{-1} \nabla_{\mathbf{w}} \mathcal{C}_{\text{val}}$
   - Compute: $\nabla^2_{\mathbf{w}} \mathcal{C}_{\text{tr}}\ \mathbf{v} = \nabla_{\mathbf{w}} \mathcal{C}_{\text{val}}$

$$\nabla_{\mathbf{x_p}} \mathcal{C}_{\text{val}} = -\nabla_{\mathbf{x_p}} \nabla_{\mathbf{w}} \mathcal{C}_{\text{tr}}\ \mathbf{v}$$

3) **Don't compute the Hessian!**

$$\frac{\partial^2 f(\mathbf{u}, \mathbf{v})}{\partial \mathbf{u} \partial \mathbf{v}^T} \mathbf{z} = \lim_{h \to 0} \frac{1}{h} \left(\nabla_{\mathbf{v}} f\left(\mathbf{u} + h\mathbf{z}, \mathbf{v}\right) - \nabla_{\mathbf{v}} f\left(\mathbf{u}, \mathbf{v}\right) \right)$$

$$\frac{\partial^2 f(\mathbf{u}, \mathbf{v})}{\partial \mathbf{u} \partial \mathbf{u}^T} \mathbf{z} = \lim_{h \to 0} \frac{1}{h} \left(\nabla_{\mathbf{u}} f\left(\mathbf{u} + h\mathbf{z}, \mathbf{v}\right) - \nabla_{\mathbf{u}} f\left(\mathbf{u}, \mathbf{v}\right) \right)$$

Poisoning with Back-Gradient Optimization

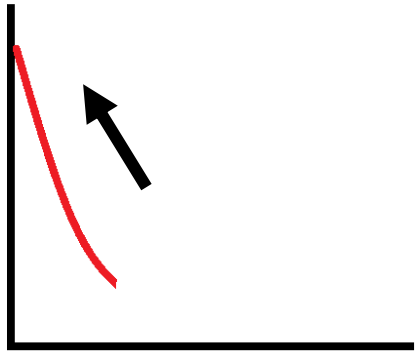
- L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E.C. Lupu, F. Roli. "*Towards Poisoning if Deep Learning Algorithms with Back-gradient Optimization.*" AISec, 2017.

Algorithm 1 Gradient Descent

Input: initial weights $\mathbf{w}_0$, learning rate α, $\mathcal{D}_{\text{tr}}$, loss function $\mathcal{L}(\mathbf{w}, \mathbf{x}, y)$

1: **for** $t = 0, \ldots, T-1$ **do**
2: $\quad \mathbf{g}_t = \nabla_{\mathbf{w}} \mathcal{C}_{\text{tr}}(\mathbf{w}_t)$
3: $\quad \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \alpha\, \mathbf{g}_t$
4: **end for**

Output: trained parameters $\mathbf{w}_T$



Algorithm 2 Back-gradient Descent

Input: $\mathbf{w}_T$, α, $\mathcal{L}(\mathbf{w}, \mathbf{x}, y)$, $\mathcal{D}_{\text{tr}}$, $\mathcal{D}_{\text{val}}$

initialize $d\mathbf{x_p} \leftarrow \mathbf{0}$, $d\mathbf{w} \leftarrow \nabla_{\mathbf{w}} \mathcal{C}_{\text{val}}(\mathbf{w}_T)$

1: **for** $t = T, \ldots, 1$ **do**
2: $\quad d\mathbf{x_p} \leftarrow d\mathbf{x_p} - \alpha\, d\mathbf{w} \nabla_{\mathbf{w}} \nabla_{\mathbf{x_p}} \mathcal{C}_{\text{tr}}(\mathbf{w}_t, \mathbf{x_p})$
3: $\quad d\mathbf{w} \leftarrow d\mathbf{w} - \alpha\, d\mathbf{w} \nabla_{\mathbf{w}} \nabla_{\mathbf{w}} \mathcal{C}_{\text{tr}}(\mathbf{w}_t, \mathbf{x_p})$
4: $\quad \mathbf{g}_{t-1} = \nabla_{\mathbf{w}_t} \mathcal{C}_{\text{tr}}(\mathbf{w}_t, \mathbf{x_p})$
5: $\quad \mathbf{w}_{t-1} = \mathbf{w}_t + \alpha\, \mathbf{g}_{t-1}$
6: **end for**

Output: $\nabla_{\mathbf{x_p}} \mathcal{C}_{\text{val}} \leftarrow d\mathbf{x_p}$

Greedy Attack Strategy

Algorithm 3 Greedy Poisoning Attack

Input: $\mathcal{D}_{\text{tr}}$, $\mathcal{D}_{\text{val}}$, iterations gradient descent T, set of initial poisoning points $\{\mathbf{x}_{p_j}^{(0)}, y_{p_j}\}_{j=0}^{n_p}$, grad. ascent learning rate β, small positive constant ε

initialize $\mathcal{D}_p \leftarrow \{\emptyset\}$, $\hat{\mathcal{D}}_{\text{tr}} \leftarrow \mathcal{D}_{\text{tr}}$

for $j = 1, \ldots, n_p$ **do**
  $i \leftarrow 0$
  repeat
    $\mathbf{w}_T \leftarrow$ Gradient Descent on $\hat{\mathcal{D}}_{\text{tr}}$ (T iterations)
    $\nabla_{\mathbf{x}_{p_j}} \mathcal{C}_{\text{val}}(\mathbf{x}_{p_j}^{(i)}, y_{p_j}) \leftarrow$ back-grad. descent with $\mathbf{w}_T$ (Alg. 2)
    $\mathbf{x}_{\mathbf{p}j}^{(i+1)} \leftarrow \Pi_{\mathcal{X}}(\mathbf{x}_{\mathbf{p}j}^{(i)} + \beta \nabla_{\mathbf{x}_{p_j}} \mathcal{C}_{\text{val}})$
    $i \leftarrow i + 1$
  until $\mathcal{C}_{\text{val}}(\mathbf{x}_{\mathbf{p}j}^{(i)}) - \mathcal{C}_{\text{val}}(\mathbf{x}_{\mathbf{p}j}^{(i-1)}) < \varepsilon$
  $\hat{\mathcal{D}}_{\text{tr}} \leftarrow \hat{\mathcal{D}}_{\text{tr}} \cup (\mathbf{x}_{p_j}^{(i)}, y_{p_j})$
  $\mathcal{D}_p \leftarrow \mathcal{D}_p \cup (\mathbf{x}_{p_j}^{(i)}, y_{p_j})$
end for

Output: set of poisoning points $\mathcal{D}_p$

- Learn one poisoning point at a time.
- Performance comparable to coordinated attack strategies.

Types of Poisoning Attacks

$$\mathcal{D}_p^* \in \arg\max_{\mathcal{D}_p} \; \mathcal{A}_{\text{val}}(\mathbf{w}^*, \mathcal{D}_{\text{val}}) \,,$$

$$\text{s.t.} \quad \mathbf{w}^* \in \arg\min_{\mathbf{w}} \; \mathcal{C}_{\text{tr}}(\mathbf{w}, \mathcal{D}_{\text{tr}} \cup \mathcal{D}_p)$$

Attacker's Objective:

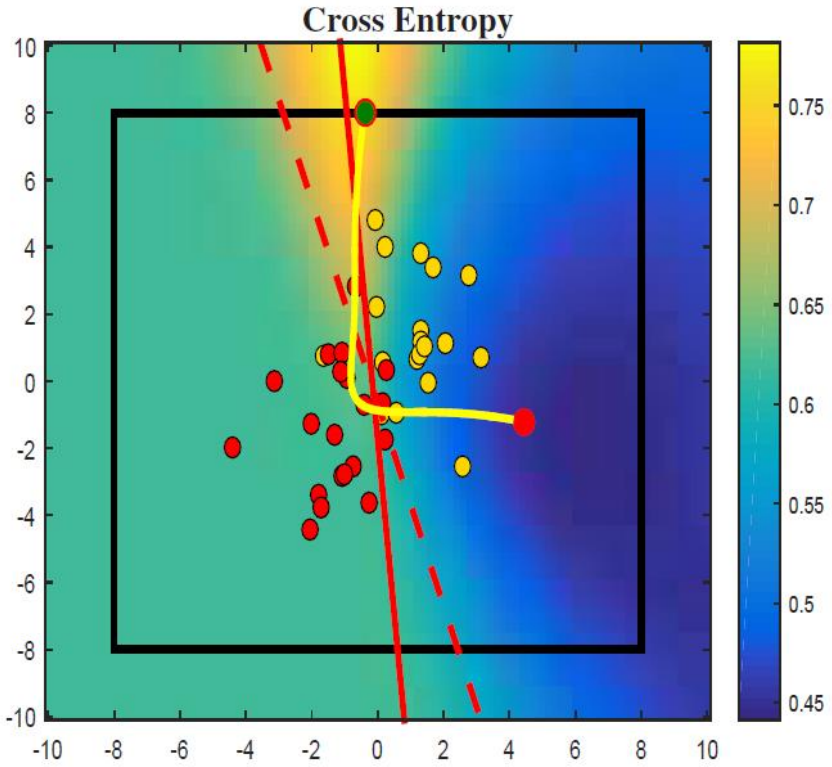
The attacker's cost function and the target samples determine the objective of the attack:

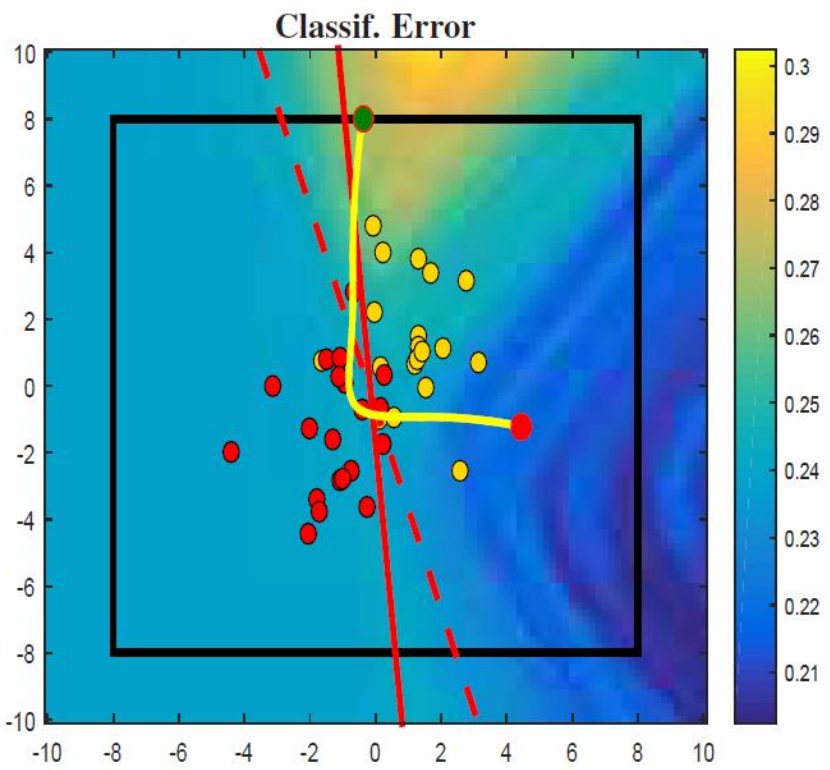
- **Targeted Attacks**: the attacker aims to cause some concrete error: particular classes, instances or features to be selected/discarded by the learning algorithm.
- **Indiscriminate Attacks**: the attacker aims to increase the overall classification error.

Attacker's Capabilities:

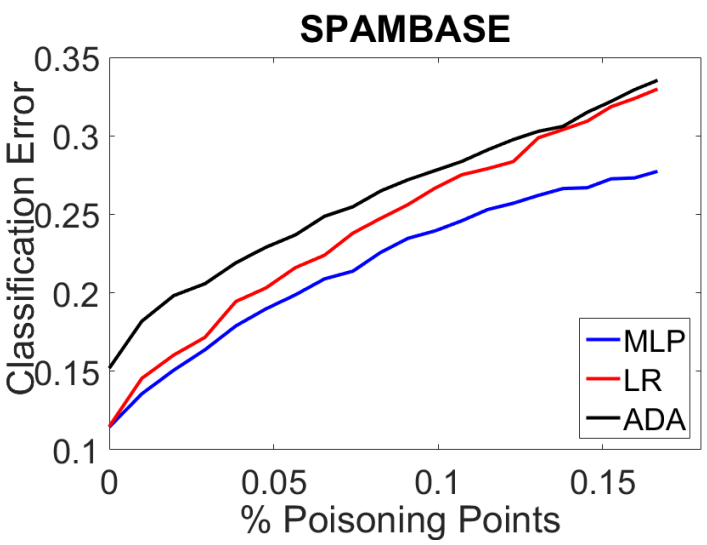
- The **labels** of the poisoning points y_p determine the attacker capabilities.
- Different modes: **insertion**, modification, deletion.
- Attacker's capabilities also have an **impact on the attacker objective**.
- **Full knowledge** vs Partial Knowledge.

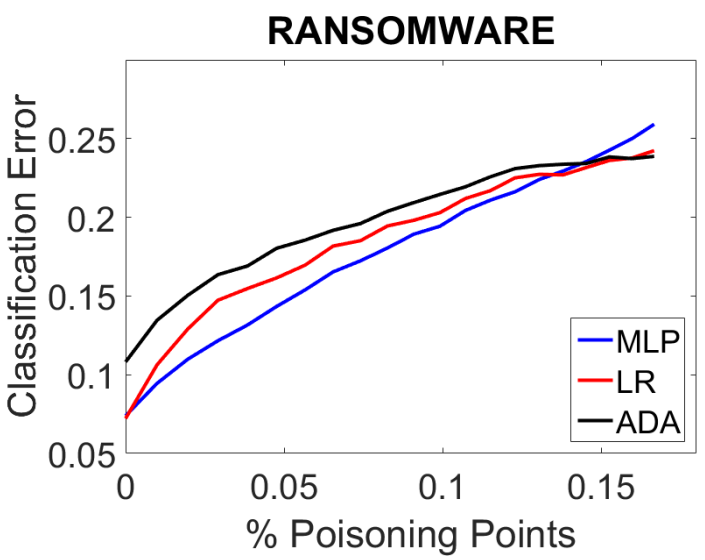
Synthetic Example

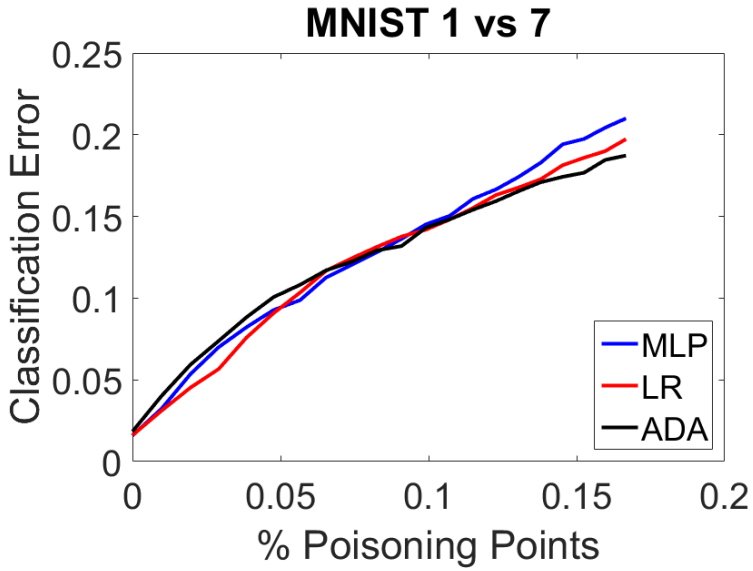




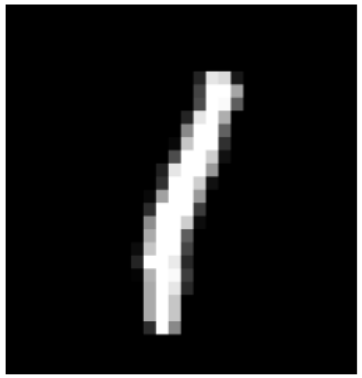
Indiscriminate Attacks against Binary Classifiers







- **Spambase**: Spam filtering application (54 features)
- **Ransomware**: Malware detection (400 features)
- **MNIST 1vs 7**: Computer vision (784 features, 28 x 28)

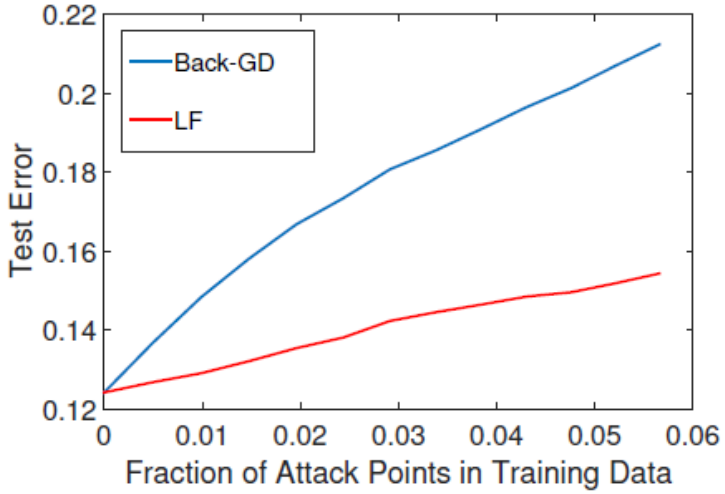


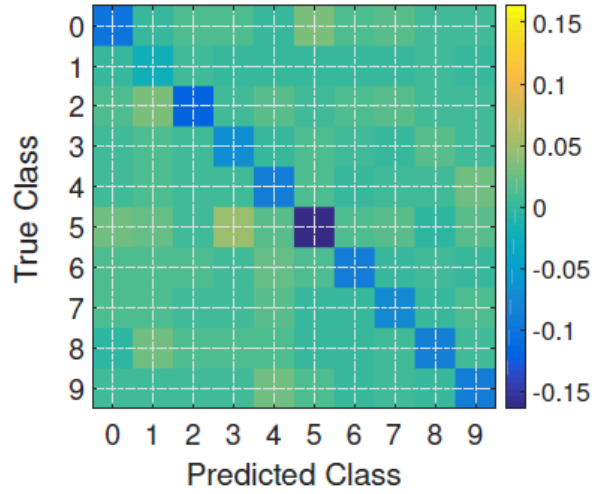
Attacks against Multi-class Classifiers

MNIST dataset (handwritten digit recognition)

Multi-class logistic Regression

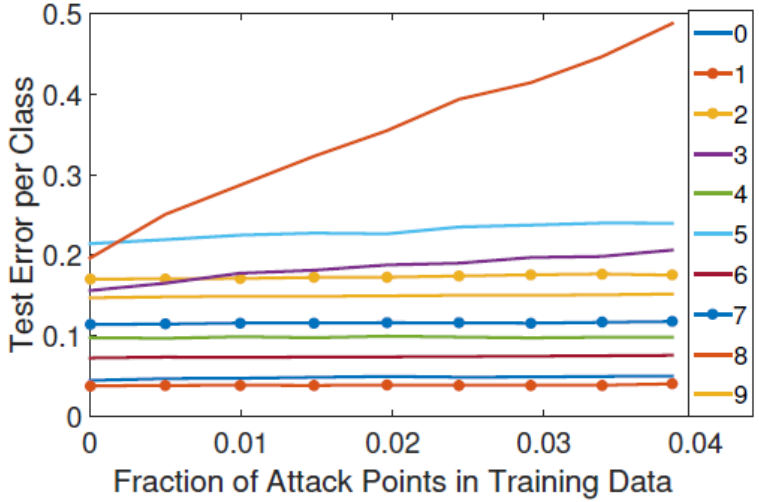
Error Generic Attack

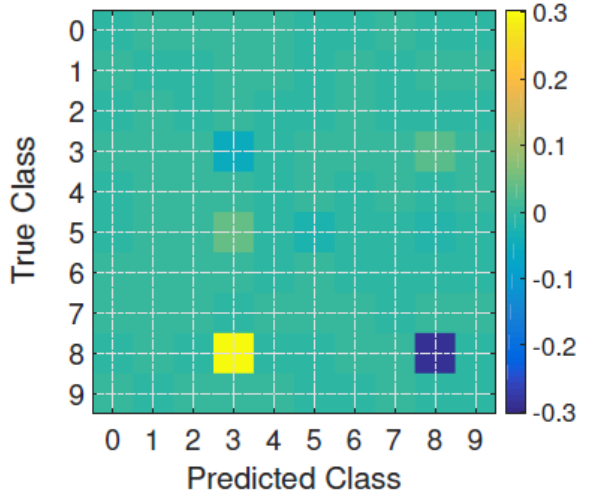




Error Specific Attack

(Misclassify digit 8 as 3)

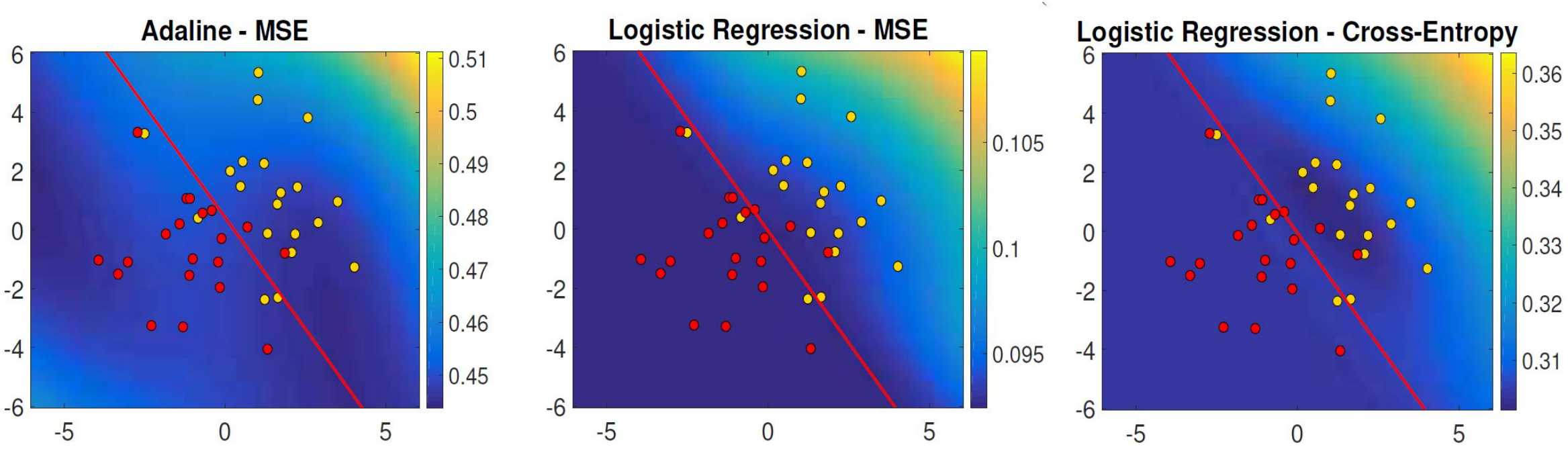




Enabling Black-Box Attacks…

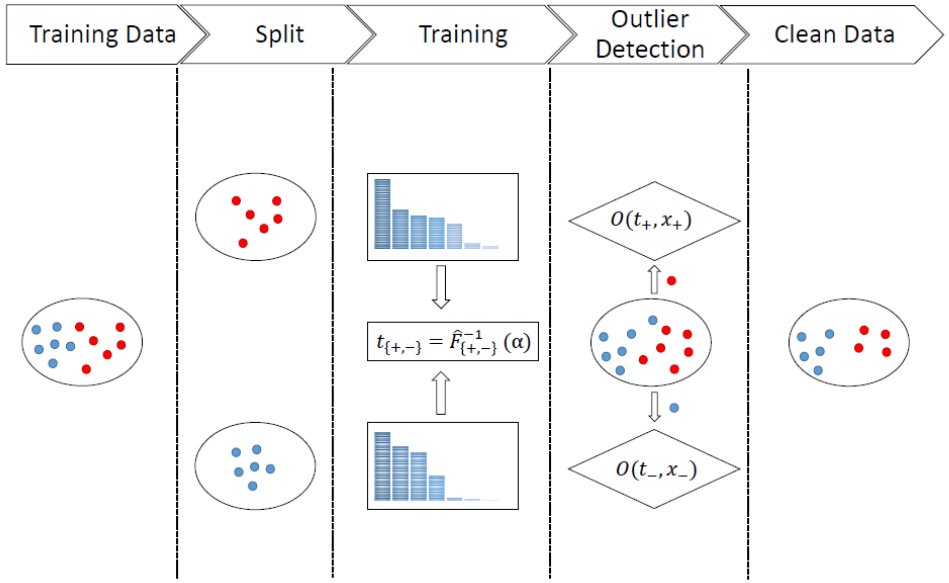
Transferability!

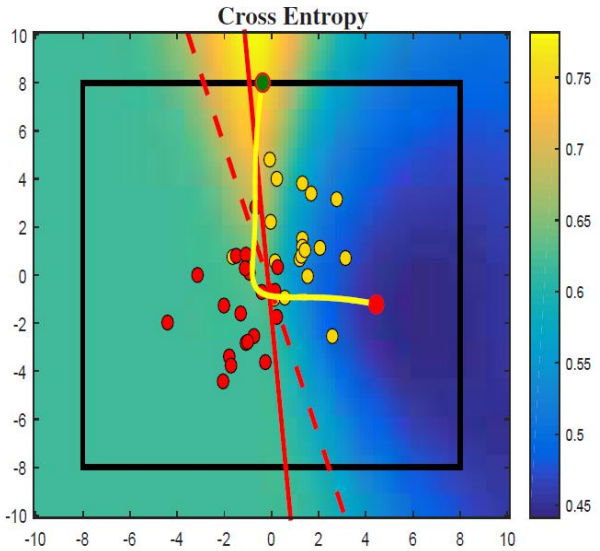
Poisoning attacks that are harmful against a targeted machine learning system are often harmful against similar systems.



Mitigation of Poisoning Attacks through Anomaly Detection

- If no adequate detectability constraints are defined the poisoning points can be far from the distribution of genuine points, i.e. can be considered as **outliers**.
- We can built **effective outlier detectors** relying on a set of curated (trusted) data points.

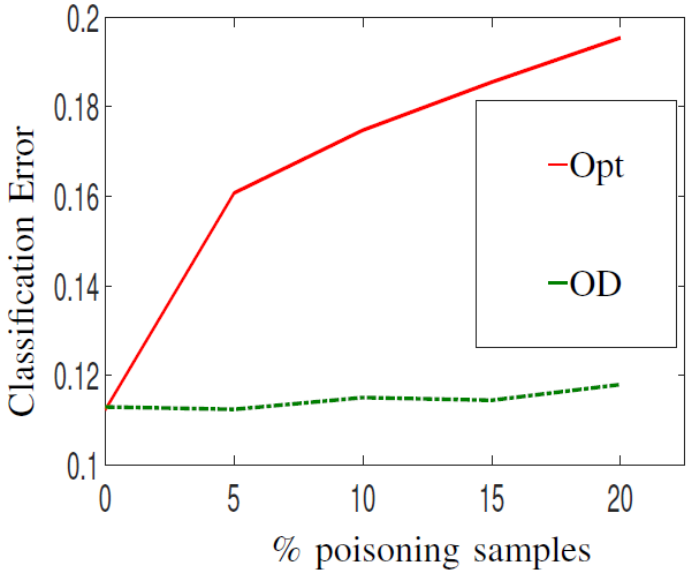




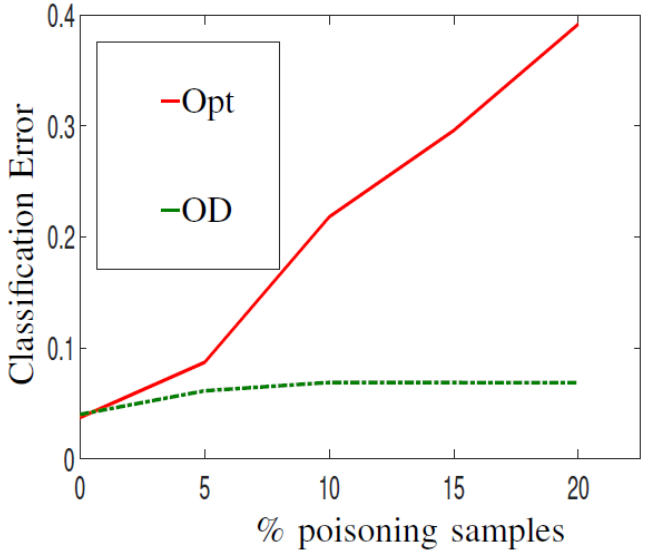
A. Paudice, L. Muñoz-González, A. Gyorgy, E.C. Lupu. "*Detection of Adversarial Training Examples in Poisoning Attacks through Anomaly Detection*." IEEE Trans. on Cybernetics (under review), 2018.

Mitigation of Poisoning Attacks through Anomaly Detection

Spambase

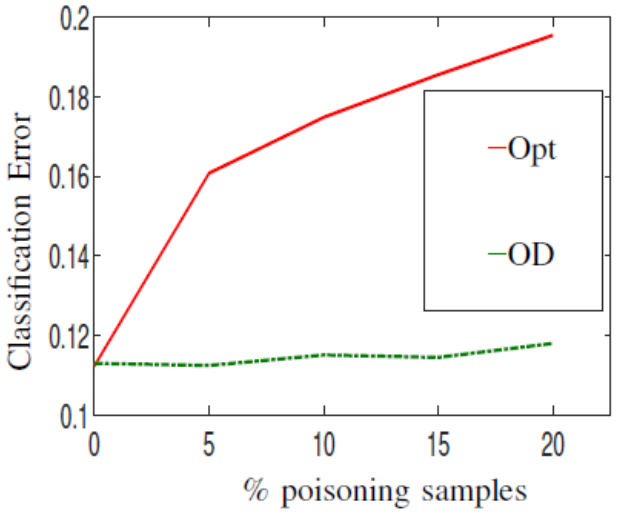


MNIST

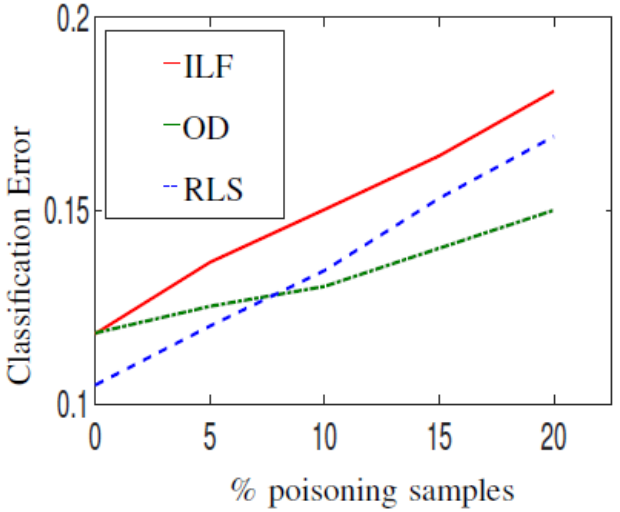


Opt: Optimal Poisoning Attack Strategy
OD: Outlier Detector Defence

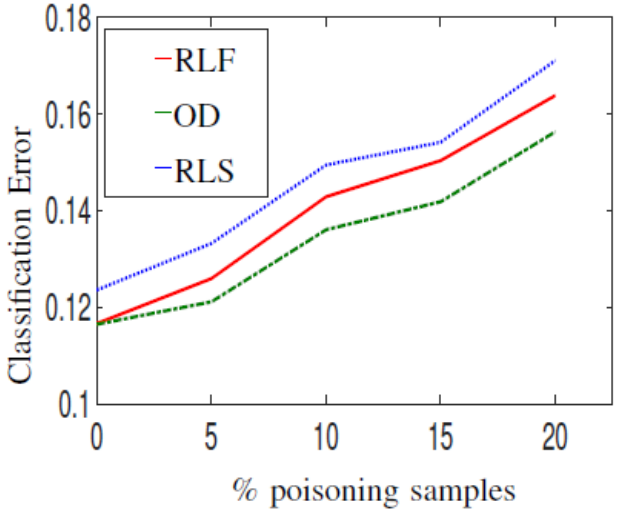
But Label Flipping Attacks are Difficult to Detect…



(a) Optimal attack (Spambase)



(b) ILF (Spambase)



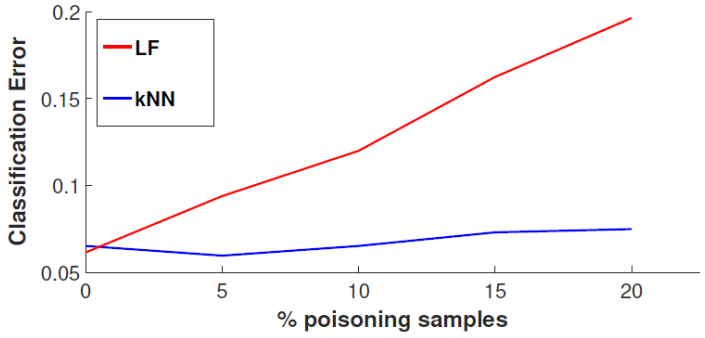
(c) RLF (Spambase)

Opt: Optimal Poisoning Attack Strategy
OD: Outlier Detector Defence
ILF: Informed Label Flipping Attack
RLF: Random Label Flipping Attack

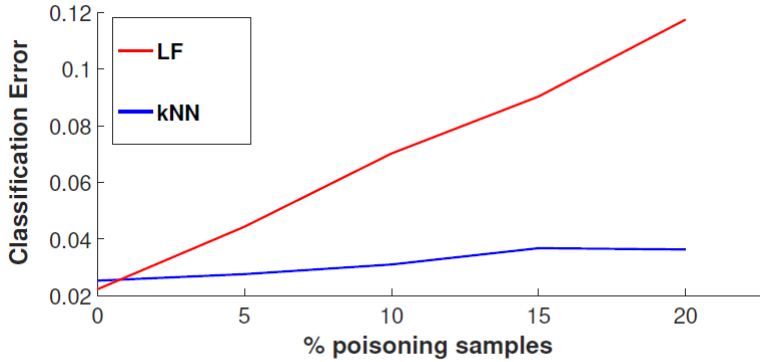
RLS: Defence from N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "*Learning with noisy labels*." NIPS, 2013

Label Sanitization to Mitigate Label Flipping Attacks

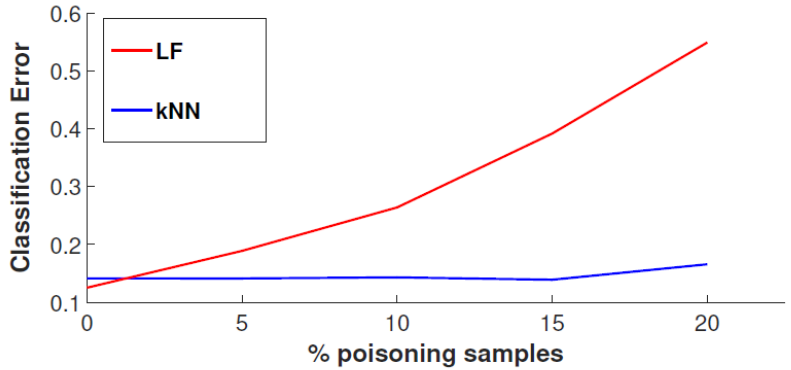
- We proposed a relabelling technique based on **K-NN** to mitigate the effect of label flipping attacks.
- But a bit of accuracy is sacrificed when the system is not under attack.



(a) BreastCancer



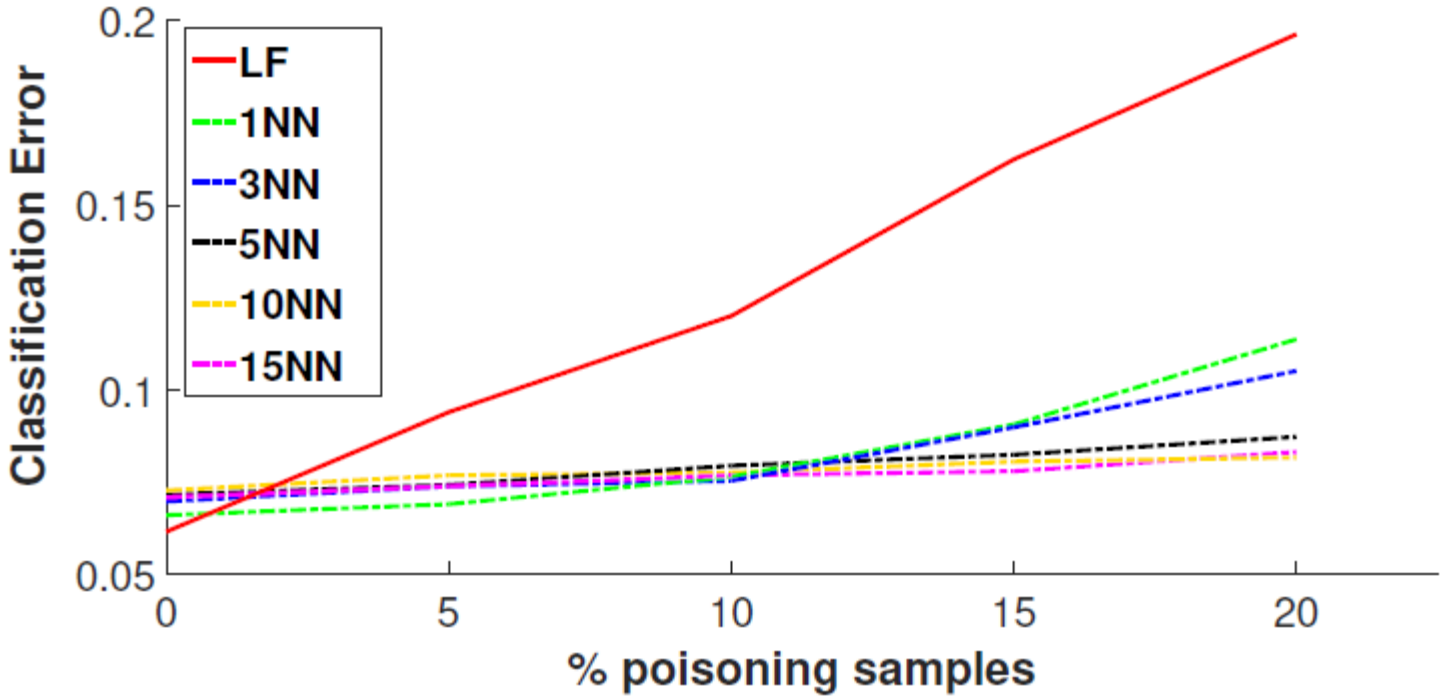
(b) MNIST



(c) (Spambase)

Label Sanitization to Mitigate Label Flipping Attacks

Sensitivity with respect to the number of neighbours:
Confidence score: 0.5



Defending against Poisoning Attacks in Online Learning Settings

- Optimal poisoning attacks in **online learning** can be modelled as a "**single-level**" optimization problem.

$$\mathbf{x}_p^* \in \arg\max_{\mathbf{x}_p \in \phi(\mathbf{x}_p)} \mathcal{A}(\mathcal{D}_t, \mathbf{w}_{n+1})$$

- We can solve the problem with a gradient ascent strategy:

$$\mathbf{x}_{p_{n+1}} = \mathbf{x}_{p_n} + \alpha \nabla_{\mathbf{x}_{p_n}} \mathcal{A}(\mathcal{D}_t, \mathbf{w}_{n+1})$$

$$\nabla_{\mathbf{x}_{p_n}} \mathcal{A}(\mathcal{D}_t, \mathbf{w}_{n+1}) = -\eta \nabla_{\mathbf{x}_{p_n}} \nabla_{\mathbf{w}_n} \mathcal{L}(\mathbf{x}_p, \mathbf{w}_n) \nabla_{\mathbf{w}_{n+1}} \mathcal{A}(\mathcal{D}_t, \mathbf{w}_{n+1})$$

- We proposed a **compounding strategy** to avoid overfitting.

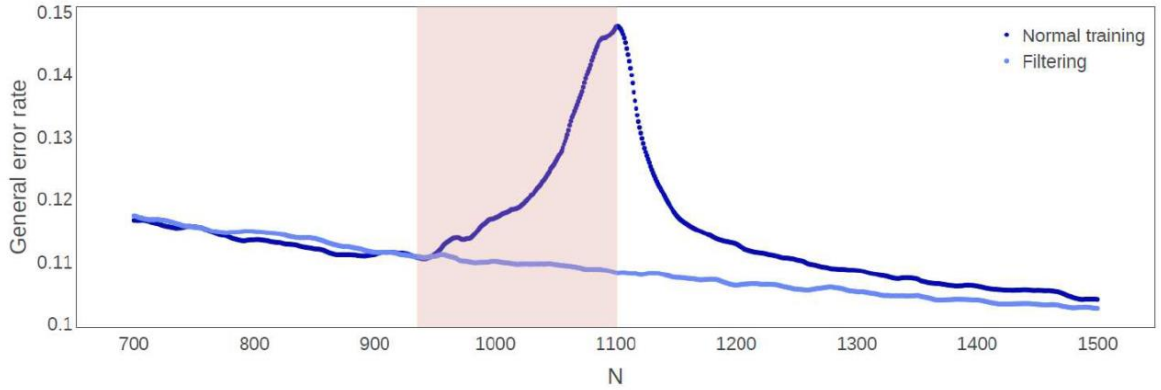
G. Collinge, E.C. Lupu, L. Muñoz-González. "*Defending against Poisoning Attacks in Online Learning Settings*." ESANN (under review), 2018.

Defending with Rejection

- In the incoming flow of data we can **reject** samples that can have a **negative impact** in the system (we don't include them for training).

$$f_{y_n}(\mathbf{x}_n, \mathbf{w}_{n-1}) < \beta$$

- Even if genuine examples are filtered out, the overall performance is not affected.

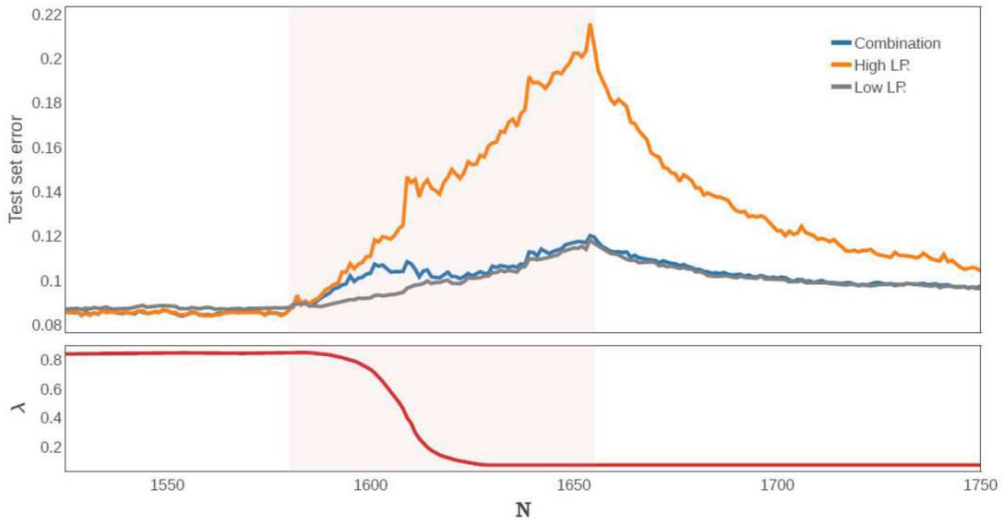


Defending with an Adaptive Combination of ML algortihms

- We use **adaptive combination** of machine learning algorithms with different learning rates.

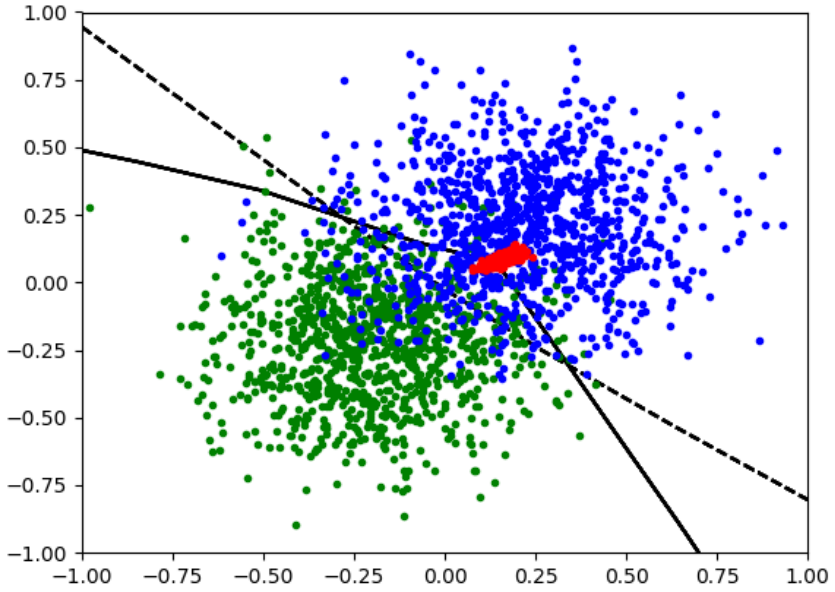
$$f_{n-1}(\mathbf{x}_n) = \lambda_{n-1} f^{(1)}_{n-1}(\mathbf{x}_n) + (1 - \lambda_{n-1}) f^{(2)}_{n-1}(\mathbf{x}_n), \qquad 0 < \lambda_{n-1} < 1$$

$$\lambda_n = \lambda_{n-1} - \alpha \nabla_{\lambda_{n-1}} \mathcal{L}(\mathcal{D}_{val}, \mathbf{w}^{(1)}_{n-1}, \mathbf{w}^{(2)}_{n-1}, \lambda_{n-1})$$



Smarter Poisoning Attacks

- Less aggressive attack strategies can’t be mitigated with outlier detection.
- We can still craft effective poisoning attacks considering detectability constraints.



Summary

- Machine learning algorithms are **vulnerable to data poisoning**.
- Optimal Poisoning Attacks can be modelled as **bi-level optimization problems**.
- **Back-Gradient optimization** is efficient to compute poisoning points:
  - Better scalability
  - No stability conditions required: can be applied to a broader range of algorithms
- **Transferability**: poisoning points generated to attack one particular algorithm can also be harmful to other algorithms.
- Interrelation between the **attacker's capabilities and objective**.
- We need to pay attention to **detectability constraints** to model more sophisticated attacks.

Imperial College London

RISS
Resilient Information Systems Security

www.rissgroup.org