

# Keyed Non-Parametric Hypothesis Tests

## Protecting Machine Learning From Poisoning Attacks

Yao Cheng<sup>1</sup>, Cheng-Kang Chu<sup>1</sup>, Hsiao-Ying Lin<sup>1</sup>,  
Marius Lombard-Platet<sup>2,3</sup>, and David Naccache<sup>2</sup>

<sup>1</sup> Huawei International, Singapore

{chengyao101, chu.cheng.kang, lin.hsiao.ying}@huawei.com

<sup>2</sup> DIENS, École normale supérieure, CNRS, PSL Research University, Paris, France

{marius.lombard-platet, david.naccache}@ens.fr

<sup>3</sup> Be-studys, Geneva, Switzerland

**Abstract.** The recent popularity of machine learning calls for a deeper understanding of AI security. Amongst the numerous AI threats published so far, poisoning attacks currently attract considerable attention.

In a poisoning attack the opponent partially tampers the dataset used for learning to mislead the classifier during the testing phase.

This paper proposes a new protection strategy against poisoning attacks. The technique relies on a new primitive called *keyed non-parametric hypothesis tests* allowing to evaluate *under adversarial conditions* the training input's conformance with a previously learned distribution  $\mathcal{D}$ . To do so we use a secret key  $\kappa$  unknown to the opponent.

Keyed non-parametric hypothesis tests differs from classical tests in that the secrecy of  $\kappa$  prevents the opponent from misleading the keyed test into concluding that a (significantly) tampered dataset belongs to  $\mathcal{D}$ .

**Keywords:** Poisoning · Machine learning security · Hypothesis tests

## 1 Introduction & Formalism

The recent popularity of machine learning calls for a deeper understanding of AI security. Amongst the numerous AI threats published so far, poisoning attacks currently attract considerable attention.

An ML algorithm  $\mathcal{A}$  is a state machine with a two-phase life-cycle: during the first phase, called *training*,  $\mathcal{A}$  builds a *model* (captured by a state variable  $\sigma_i$ ) based on sample data  $D$ , called “training data”:

$$D = \{d_1, \dots, d_k\} \text{ where } d_i = \{\text{data}_i, \text{label}_i\}$$

Learning is hence defined by:

$$\sigma_i \leftarrow \mathcal{A}(\text{learn}, \sigma_{i-1}, d_i)$$

e.g.  $\text{data}_i$  can be human face images and the  $\text{label}_i \in \{\sigma, \varphi\}$ .

During the second phase, called *testing*<sup>1</sup>,  $\mathcal{A}$  is given an unlabelled data.  $\mathcal{A}$ 's goal is to predict as accurately as possible the corresponding label given the distribution  $\mathcal{D}$  inferred from  $D$ .

$$\underline{\text{label}} = \mathcal{A}(\text{test}, \sigma_k, \text{data})$$

We denote by  $T$  the dataset  $\{\text{data}_i, \text{label}_i\}$  used during testing where  $\text{label}_i$  is the correct label (solution) corresponding to  $\text{data}_i$  and  $\underline{\text{label}}_i$  is the label predicted by  $\mathcal{A}(\text{test}, \sigma_k, \text{data}_i)$ <sup>2</sup>.

In a poisoning attack the opponent partially tampers  $D$  to influence  $\sigma_k$  and mislead  $\mathcal{A}$  during testing. Formally, letting  $\bar{d} = \{\overline{\text{data}}, \overline{\text{label}}\}$ , the attacker generates a *poison* dataset

$$\tilde{D} = \{\tilde{d}_1, \dots, \tilde{d}_k\}$$

resulting in a corrupted model  $\tilde{\sigma}_k$  such that

$$\overline{\text{label}} \neq \underline{\text{label}} = \mathcal{A}(\text{test}, \tilde{\sigma}_k, \overline{\text{data}})$$

Poisoning attacks were successfully implemented by tampering both incremental and periodic training models. In the *incremental training model*<sup>3</sup>, whenever a new  $d_i$  is seen during testing,  $\mathcal{A}$ 's performance on  $d_i$  is evaluated and  $\sigma$  is updated. In the *periodic retraining model*, data is stored in a buffer. When  $\mathcal{A}$  falls below a performance threshold (or after a fixed number of queries) the buffer's data is used to retrain  $\mathcal{A}$  anew. Retraining is either done using the buffer alone (resulting in a totally new  $\sigma$ ) or by merging the buffer with previous information (updating  $\sigma$ ).

Protections against poisoning attacks can be categorized into two types: *robustification* and *sanitizing*:

**Robustification** (built-in resistance) modifies  $\mathcal{A}$  so that it takes into account the poison but tolerates its effect. Note that  $\mathcal{A}$  does not need to identify the poisoned data as such but the effect of poisonous data must be diminished, dampened or nullified up to a point fit for purpose.

The two main robustification techniques discussed in the literature are:

*Feature squeezing* [32,26] is a model hardening technique that reduces data complexity so that adversarial perturbations disappear because of low sensitivity. Usually the quality of the incoming data is degraded by encoding colors with fewer values or by using a smoothing filter over the images. This maps several inputs onto one "characteristic" or "canonical input" and reduces the perturbations introduced by the attacker. While useful in practice, those techniques inevitably degrade the  $\mathcal{A}$ 's accuracy.

<sup>1</sup> Or *inference*.

<sup>2</sup> i.e. if  $\mathcal{A}$  is perfect then  $\underline{\text{label}} = \text{label}$ .

<sup>3</sup> Also called the *incremental update model*.

*Defense-GANs* [25] use Generative Adversarial Networks [10] to reduce the poison’s efficiency. Informally, the GAN builds a model of the learned data and projects the input onto it.

**Sanitizing** detects (by various methods e.g. [8,17]) and discards poisoned  $d_i$ s. Note that sanitizing necessarily decreases  $\mathcal{A}$ ’s ability to learn.

This work prevents poisoning by sanitizing.

Figure 1 shows a generic abstraction of sanitizing.  $\mathcal{A}$  takes  $D$  (periodically or incrementally) and outputs a  $\sigma$  for the testing phase. But  $d_i$ s go through the poisoning detection module  $\text{Det}$  before entering  $\mathcal{A}$ . If  $\text{Det}$  decides that the probability that some  $d_i$  is poisoned is too high, the suspicious  $d_i$  is trashed to avoid corrupting  $\sigma$ .

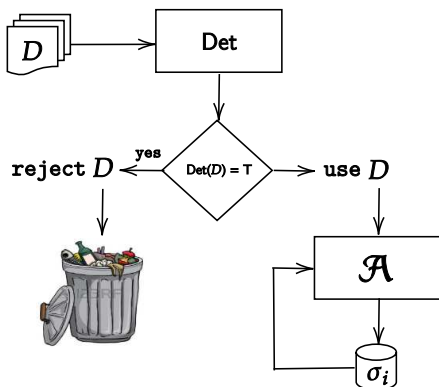


Fig. 1: Before entering  $\mathcal{A}$ ,  $D$  is given to the poison detection module  $\text{Det}$ . If  $\text{Det}$  decides that  $D$  is poisoned,  $D$  is trashed. Otherwise  $D$  is fed into  $\mathcal{A}$  who updates  $\sigma$ .

Because under normal circumstances  $D$  and  $T$  are drawn from the same distribution  $\mathfrak{D}$  it is natural to implement  $\text{Det}$  using standard algorithms allowing to test the hypothesis  $D \in \mathfrak{D}$ .

The most natural tool allowing one to do so is *nonparametric hypothesis tests* (NPHTs, hereafter denoted by  $G$ ). Let  $A, B$  be two datasets.  $G(A, B) \in \{\text{T}, \text{F}\}$  allows to judge how compatible is a difference observed between  $A$  and  $B$  with the hypothesis that  $A, B$  were drawn from the same distribution  $\mathfrak{D}$ .

It is important to underline that  $G$  is nonparametric, i.e.  $G$  makes no assumptions on  $\mathfrak{D}$ .

The above makes NPHTs natural candidates for detecting poison. However, whilst NPHTs are very good for natural hypothesis testing, they succumb spec-

tacularly in adversarial scenarios where the attacker has full knowledge of the target’s specification [13]. Indeed, section 3.1 illustrates such a collapse.

To regain a head-up over the attacker, our strategy will consist in mapping  $A$  and  $B$  into a *secret* space unpredictable by the adversary where  $G$  can work confidentially. This mapping is defined by a key  $\kappa$  making it hard for the adversary to design  $A \in \mathfrak{D}$  and  $B \in \mathfrak{D}'$  such that

$$G(A, B) = \top \text{ and } \mathfrak{D} \neq \mathfrak{D}'$$

## 2 A Brief Overview of Poisoning Attacks

Barreno et al. [2] were the first to coin the term “poisoning attacks”. Follow-up works such as Kearns et al. [12] sophisticated and theorized this approach.

Classical references introducing poisoning are [23,22,20,24,4,30,21,31,18,7,15]. At times (e.g. [15]) the opponent does not create or modify  $d_i$ ’s but rather adds legitimate but carefully chosen  $d_i$ ’s to  $D$  to bias learning. Those inputs are usually computed using gradient descent. This was later generalized by [25].

During a poisoning attack, data modifications can either concern data or labels. [3] showed that a random flip of 40% of labels suffices to seriously affect SVMs. [14] showed that inserting malicious points into  $D$  could gradually shift the decision boundary of an anomaly detection classifier. Poisoning points were obtained by solving a linear programming problem maximizing the mean of the displacement of the mass center of  $D$ . For a more complete overview we recommend [5].

**Adversarial Goals.** Poisoning may seek to influence the classifier’s decision when presented with a later target query or to leak information about  $D$  or  $\sigma$ .

The attacker’s goals always apply to the testing phase and may be:

- *Confidence Reduction*: Have  $\mathcal{A}$  make more errors. In many cases, “less confidence” can clear suspicious instances at the benefit of doubt (*in dubio pro reo*).
- *Mis-classification attacks*: are defined by replacing  $\text{adj}_1, \text{adj}_2$  in the definition:

“Make  $\mathcal{A}$  conclude that a  $\text{adj}_1$   $\text{data}_i$  belongs to a  $\text{adj}_2$  wrong label.”

Attack	$\text{adj}_1$	$\text{adj}_2$
Mis-classification <sup>4</sup>	random	random
Targeted Mis-classification	chosen	random
Source-Target Mis-classification	chosen	chosen

<sup>4</sup> This is useful if any mistake may serve the opponent’s interests e.g. any navigation error would crash a drone with high probability.

**Adversarial capabilities** designate the degree of knowledge that the attacker has on the target system. Bibliography distinguishes between *training phase capabilities* and *testing phase capabilities*. Poisoning assumes training phase capabilities.

The attacker’s capabilities may be:

- *Data Injection*: Add new data to  $D$ .
- *Data Modification*: Modify  $D$  before training.
- *Logic Corruption*: Modify the code (behavior) of  $\mathcal{A}$ <sup>5</sup>.

### 3 Keyed Anti-Poisoning

To illustrate our strategy, we use Mann-Whitney’s  $U$ -test and Stouffer’s method that we recall in the appendix.

We assume that when training starts, we are given a safe subset of  $D$  denoted  $D_s$  (where the subscript  $s$  stands for “safe”). Our goal is to assess the safety of the upcoming subset of  $D$  denoted  $D_u$  (where the subscript  $u$  stands for “unknown”).

We assume that  $D_s$  and  $T$  come from the same distribution  $\mathfrak{D}$ . As mentioned before, the idea is to map  $D_s$  and  $D_u$  to a space  $f_\kappa(\mathfrak{D})$  hidden from the opponent.  $f$  is keyed to prevent the attacker from predicting how to create adversarial input fooling  $\mathcal{A}$ .

Figure 2 shows the details of the Det plugin added to  $\mathcal{A}$  in Figure 1. Det takes a key  $\kappa$ , reads  $D_s, D_u$ , performs the keyed transformation, calls  $G$  on  $f_\kappa(D_u), f_\kappa(D_s)$  and outputs a decision.

$G$  can be Mann-Whitney’s test (illustrated in this paper) or any other NPHT e.g. the location test, the paired  $T$  test, Siegel-Turkey’s test, the variance test, or multidimensional tests such as deep gaussianization [29].

#### 3.1 Trivial Mann-Whitney Poisoning

Let  $G$  be Mann-Whitney’s  $U$ -Test returning a  $p$ -value:

$$p = G(A, B) = G(\{a_0, \dots, a_{n_1-1}\}, \{b_0, \dots, b_{n_2-1}\})$$

$G$  is, between others, susceptible to poisoning as follows: assume that  $A$  is sampled from a Gaussian distribution  $A \in_R \mathcal{N}(\mu_A, \sigma_A)$  and that  $B$  is sampled from  $\{-q, q\}$  where  $\mu_A \ll q$  (Figure 3). While  $A$  and  $B$  are totally different,  $G$  will be misled.

For instance, after picking  $10^6$  samples  $A \in_R \mathcal{N}(0, 3)$  and  $10^6$  samples  $B \in_R \{-15, 15\}$  (i.e. we took  $q = 15$ ), we get a  $p$ -value of 0.99. From Mann-Whitney’s perspective,  $A, B$  come from the same parent distribution with a very high degree of confidence while, in all evidence, they do not.

<sup>5</sup> This is the equivalent of fault attacks in cryptography.

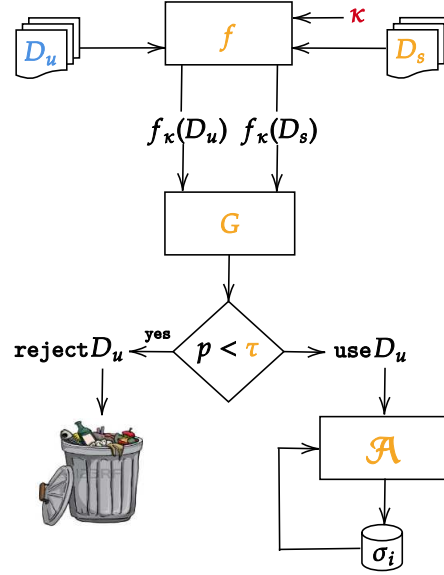


Fig. 2: Implementing Det using keying. **Red**: unknown to the opponent. **Orange**: known by the opponent. **Blue**: controlled by the opponent.

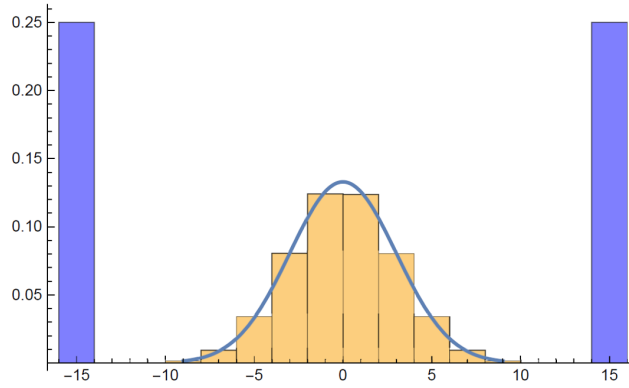


Fig. 3: Trivial Mann-Whitney poisoning. Samples drawn from the blue distribution are Mann-Whitney-indistinguishable from samples drawn from the orange one.

### 3.2 Keyed Mann-Whitney

We instantiate  $f_\kappa$  by secret random polynomials i.e. polynomials  $R(x)$  whose coefficients are randomly and secretly refreshed before each invocation of  $G$ . Instead of returning  $G(A, B)$ , Det returns  $G(R(A), R(B))$  where:

$$R(A) = \{R(a_0), \dots, R(a_{n_1-1})\} \quad \text{and} \quad R(B) = \{R(b_0), \dots, R(b_{n_2-1})\}$$

The rationale is that  $R$  will map the attacker’s input to an unpredictable location in which the Mann-Whitney is very likely to be safe.

$\ell$  random polynomials  $R_1(x), \dots, R_\ell(x)$  are selected as keys and Det calls  $G$  for each polynomial. To aggregate all resulting  $p$ -values, Det computes:

$$\Delta = \text{Stouffer}\left(G\left(R_1(A), R_1(B)\right), \dots, G\left(R_\ell(A), R_\ell(B)\right)\right)$$

If  $\Delta \approx 0$ , the sample is rejected as poisonous with very high probability.

Note that any smooth function can be used as  $R$ , e.g. B-splines. The criterion on  $R$  is that the random selection process must yield significantly different functions.

### 3.3 Experiments

We illustrate the above by protecting  $\mathcal{N}(0, 1)$ . The good thing about  $\mathcal{N}(0, 1)$  is that random polynomials tend to diverge when  $x = 1$  but adapt well to the central interval in which the Gaussian is not negligible.

We attack  $\mathcal{N}(0, 1)$  by poisoning with  $\{-q, q\}$ , where  $q$  is set to 3, 2, 1, and 0.5, respectively. For each value of  $q$ , two sets of 50 samples are drawn from the two distributions. Those samples are then transformed into other sets by applying a random polynomial of degree 4 and then fed into  $G$  to obtain a  $p$ -value (using the two-sided mode). This  $p$ -value predicts whether these two sets of transformed samples come from the same distribution: a  $p$ -value close to 0 is a strong evidence against the null hypothesis. In each of our experiments, we apply nine secret random polynomials of degree 4 and aggregate the resulting  $p$ -values using Stouffer’s method. For each setting, we run 1000 simulations. Similarly, for the same polynomials and  $q$ , we run a “honest” test, where both samples come from the same distribution.

We thus retrieve 1000 “attack”  $p$ -values, which we sort by ascending order. Similarly, we sort the “honest”  $p$ -values. It is a classic result that, under the null hypothesis, a  $p$ -value follows a random uniform distribution over  $[0, 1]$ , hence a plot of the sorted “honest”  $p$ -values is a linear curve.

An attack is successful if, on average, the “attack” sample is accepted as least as often as the “honest” sample. This can be rewritten as  $E(p^{\text{attack}}) \geq E(p^{\text{honest}})$ , with  $E$  the . Hence, a sufficient condition for the validity is that the curve of sorted attack  $p$ -values (solid lines in our figures) is above the curve of sorted honest  $p$ -values (dashed lines).

Experimental results are summarized in Figures 4, 5, 6 and 7.

The first quadrant illustrates the polynomials used in the simulation and two bars for  $\{-q, q\}$ . The same random polynomials were used for each experiment. For simplicity, the coefficients of the polynomials were uniformly selected from  $\{-1, 0, 1\}$ , and (useless) polynomials of degree lower than 2 were excluded from the random selection. Then, we also added the identity polynomial (poly0), as a witness of what happens when there is no protection.

The following nine quadrants give the distribution of  $p$ -values for each polynomial, over 1000 simulations, sorted in increasing order. The dotted distribution corresponds to what an honest user would obtain, whereas the plain line simulation is based on poisoned datasets.

The last quadrant contains the sorted distribution of the aggregated  $p$ -values using Stouffer. Experimental results show that for poisoned datasets, the aggregated  $p$ -values remain close to zero, while a honest dataset does not appear to be significantly affected. In other words, with very high probability, keyed testing detects poisoning.

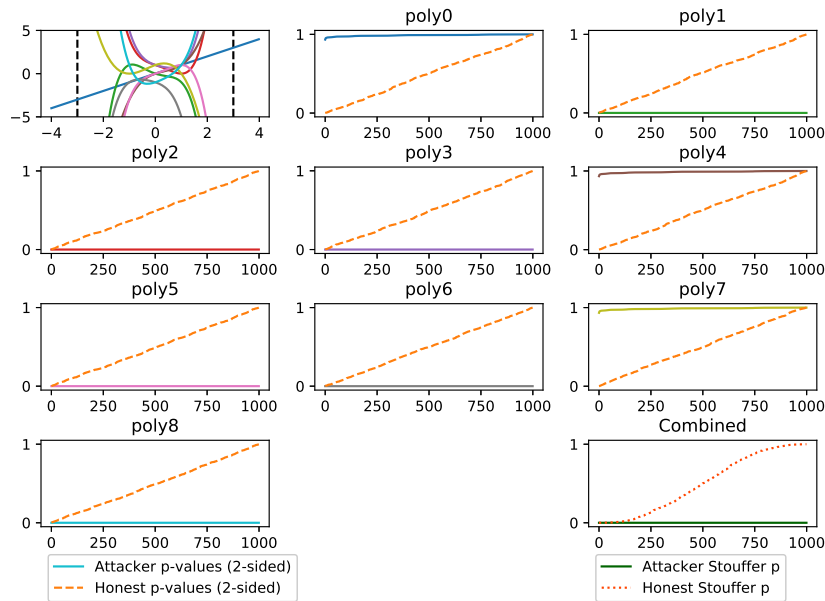
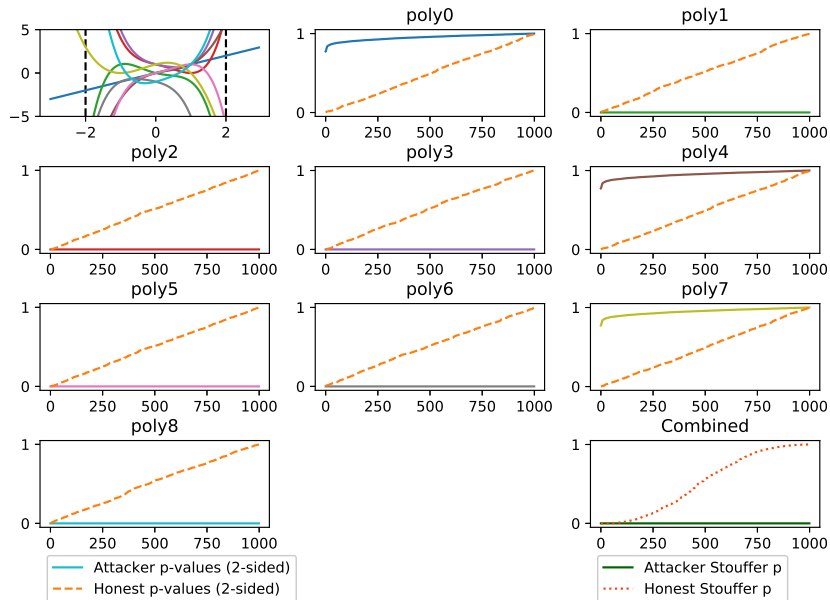
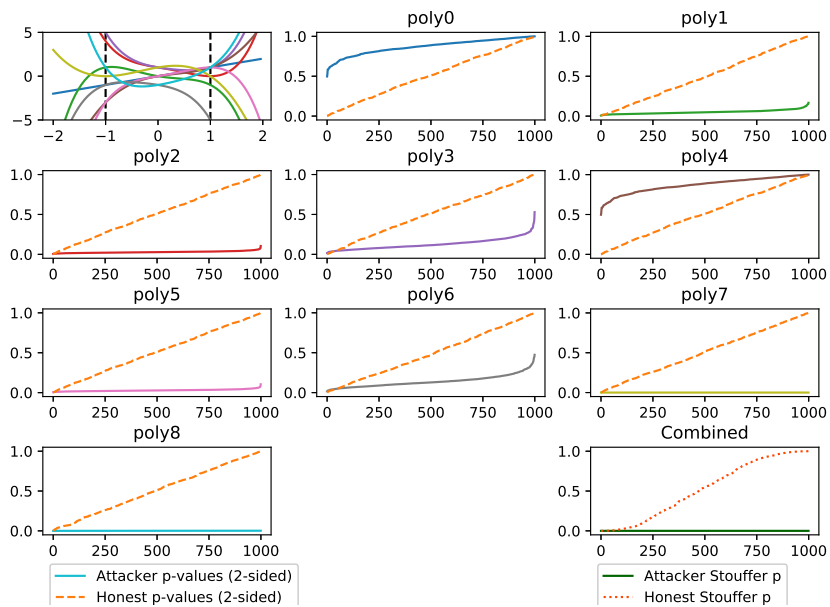


Fig. 4: Attack with  $q = 3$ . Defense with polynomials of degree 4.

### 3.4 Discussion

We observe a saturation when  $q$  is too far from  $\mu_A$ , this is due to the fact that even after passing through  $R$  the attack samples remain at the extremes. Hence



Fig. 5: Attack with  $q = 2$ . Defense with polynomials of degree 4.Fig. 6: Attack with  $q = 1$ . Defense with polynomials of degree 4.

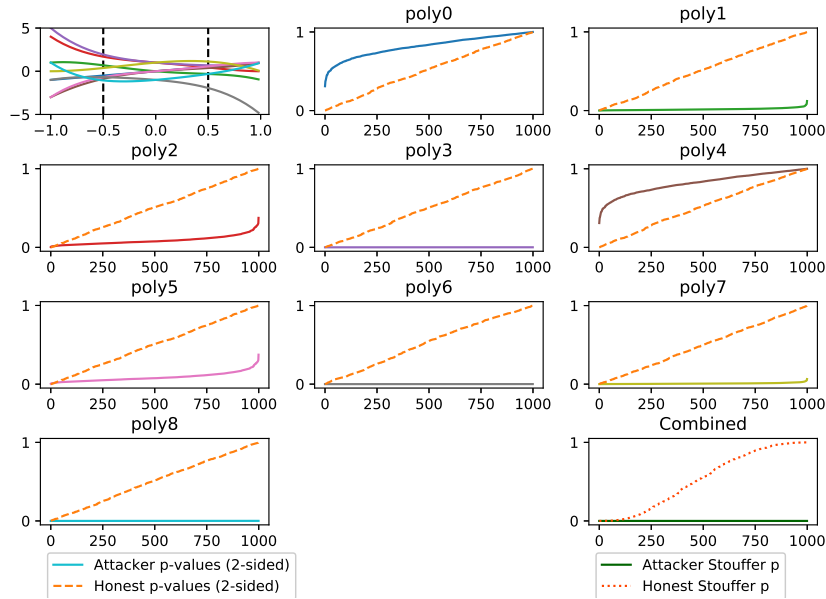


Fig. 7: Attack with  $q = 0.5$ . Defense with polynomials of degree 4.

if  $R$  is of odd degree, nothing changes. If the degree of  $R$  is even then the two extremes are projected to the same side and Mann-Whitney detects 100% of the poisonings. It follows that at saturation a keyed Mann-Whitney gives either very high or very low  $p$ -value. This means that polynomials or B-splines must be carefully chosen to make keying effective.

The advantage of combining the  $p$ -values with Stouffer’s method is that the weak  $p$ -values are very penalizing (by opposition to Pearson’s method whose combined  $p$ -value degrades much slower). A more conservative aggregation would be using Fisher’s method.

All in all, experimental results reveal that keying managed to endow the test with a significant level of immunity.

Interestingly, Det can be implemented independently of  $\mathcal{A}$ .

A cautionary note: Our scenario assumes that testing does not start before learning is complete. If the opponent can alternate learning and testing then he may infer that a poisonous sample was taken into account (if  $\sigma$  was updated and  $\mathcal{A}$ ’s behavior was modified). This may open the door to attacks on  $\mathcal{A}$ .

## 4 Notes and Further Research

This paper opens perspectives beyond the specific poisoning problem. e.g. cryptographers frequently use randomness tests  $\mathcal{R}$  to assess the quality of random number generators. In a strong attack model where the attacker knows  $\mathcal{R}$  and

controls the random source it becomes possible to trick many  $\mathcal{R}$ s into declaring flagrantly non random data as random. Hence, the authors believe that developing *keyed randomness tests*  $\mathcal{R}_\kappa$  is important and useful as such.

For instance, in the original minimum distance test 8000 points (a set  $S$ ) sampled from the tested randomness source  $\mathcal{S}$  are placed in a  $10000 \times 10000$  square. Let  $\delta$  be the minimum distance between the pairs. If  $\mathcal{S}$  is random then  $\delta^2$  is exponentially distributed with mean 0.995. To key the test a secret permutation  $R_\kappa$  of the plan can generated and the test can be applied to  $R_\kappa(S)$ .

To the best of our knowledge such primitives were not proposed yet.

We note, however, that keyed protections to different (non cryptographic!) decision problems in very diverse areas do emerge independently e.g. [19,1,9,28].

## References

1. Albrecht, M.R., Massimo, J., Paterson, K.G., Somorovsky, J.: Prime and prejudice: Primality testing under adversarial conditions. Cryptology ePrint Archive, Report 2018/749 (2018), <https://eprint.iacr.org/2018/749>
2. Barreno, M., Nelson, B., Joseph, A.D., Tygar, J.: The security of machine learning. *Machine Learning* **81**(2), 121–148 (Nov 2010)
3. Biggio, B., Nelson, B., Laskov, P.: Support vector machines under adversarial label noise. In: Hsu, C.N., Lee, W.S. (eds.) *Proceedings of the Asian Conference on Machine Learning*. *Proceedings of Machine Learning Research*, vol. 20, pp. 97–112. PMLR, South Garden Hotels and Resorts, Taoyuan, Taiwan (14–15 Nov 2011)
4. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines. In: *Proceedings of the 29th International Conference on International Conference on Machine Learning*. pp. 1467–1474. ICML’12, Omnipress, USA (2012)
5. Biggio, B., Roli, F.: Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition* **84**, 317 – 331 (2018)
6. Brown, M.: A method for combining non-independent, one-sided tests of significance. *Biometrics* **31**(4), 987–992 (1975)
7. Burkard, C., Lagesse, B.: Analysis of Causative Attacks Against SVMs Learning from Data Streams. In: *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*. pp. 31–36. IWSPA ’17, ACM, New York, NY, USA (2017)
8. Cretu, G.F., Stavrou, A., Locasto, M.E., Stolfo, S.J., Keromytis, A.D.: Casting out demons: Sanitizing training data for anomaly sensors. In: *2008 IEEE Symposium on Security and Privacy* (sp 2008). pp. 81–95 (May 2008)
9. Géraud, R., Lombard-Platet, M., Naccache, D.: Quotient hash tables - efficiently detecting duplicates in streaming data. *CoRR* **abs/1901.04358** (2019)
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: *Advances in neural information processing systems*. pp. 2672–2680 (2014)
11. Heard, N.A., Rubin-Delanchy, P.: Choosing between methods of combining-values. *Biometrika* **105**(1), 239–246 (2018)
12. Kearns, M., Li, M.: Learning in the presence of malicious errors. In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. pp. 267–280. STOC ’88, ACM, New York, NY, USA (1988)

13. Kerckhoffs, A.: La cryptographie militaire. In: Journal des sciences militaires. vol. IX, pp. 5–38 (January 1883)
14. Kloft, M., Laskov, P.: Online anomaly detection under adversarial impact. In: Teh, Y.W., Titterton, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 9, pp. 405–412. PMLR, Chia Laguna Resort, Sardinia, Italy (13–15 May 2010)
15. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. pp. 1885–1894. ICML'17, JMLR.org (2017)
16. Kost, J.T., McDermott, M.P.: Combining dependent  $p$ -values. *Statistics & Probability Letters* **60**(2), 183 – 190 (2002)
17. Laishram, R., Phoha, V.V.: Curie: A method for protecting SVM Classifier from Poisoning Attack. ArXiv [abs/1606.01584](https://arxiv.org/abs/1606.01584) (2016)
18. Mei, S., Zhu, X.: Using machine teaching to identify optimal training-set attacks on machine learners. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. pp. 2871–2877. AAAI'15, AAAI Press (2015)
19. Naor, M., Yogev, E.: Bloom filters in adversarial environments. CoRR [abs/1412.8356](https://arxiv.org/abs/1412.8356) (2014)
20. Nelson, B., Barreno, M., Chi, F.J., Joseph, A.D., Rubinstein, B.I., Saini, U., Sutton, C., Tygar, J., Xia, K.: Exploiting machine learning to subvert your spam filter. In: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats. pp. 7:1–7:9. LEET'08, USENIX Association, Berkeley, CA, USA (2008)
21. Newell, A., Potharaju, R., Xiang, L., Nita-Rotaru, C.: On the practicality of integrity attacks on document-level sentiment analysis. In: Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop. pp. 83–93. AISec'14, ACM, New York, NY, USA (2014)
22. Newsome, J., Karp, B., Song, D.: Paragraph: Thwarting signature learning by training maliciously. In: Proceedings of the 9th International Conference on Recent Advances in Intrusion Detection. pp. 81–105. RAID'06, Springer-Verlag, Berlin, Heidelberg (2006)
23. Perdisci, R., Dagon, D., Wenke Lee, Fogla, P., Sharif, M.: Misleading worm signature generators using deliberate noise injection. In: 2006 IEEE Symposium on Security and Privacy (S&P'06). pp. 15 pp.–31 (May 2006)
24. Rubinstein, B.I., Nelson, B., Huang, L., Joseph, A.D., Lau, S.h., Rao, S., Taft, N., Tygar, J.D.: ANTIDOTE: Understanding and Defending Against Poisoning of Anomaly Detectors. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement. pp. 1–14. IMC '09, ACM, New York, NY, USA (2009)
25. Samangouei, P., Kabkab, M., Chellappa, R.: Defense-GAN: Protecting classifiers against adversarial attacks using generative models. arXiv preprint [arXiv:1805.06605](https://arxiv.org/abs/1805.06605) (2018)
26. Shaham, U., Garritano, J., Yamada, Y., Weinberger, E., Cloninger, A., Cheng, X., Stanton, K.P., Kluger, Y.: Defending against adversarial images using basis functions transformations. ArXiv [abs/1803.10840](https://arxiv.org/abs/1803.10840) (2018)
27. Stouffer, S., Suchman, E., DeVinney, L., Star, S., Williams, R.J.: *The American Soldier, Vol.1: Adjustment during Army Life.* (1949)
28. Taran, O., Rezaeifar, S., Voloshynovskiy, S.: Bridging machine learning and cryptography in defence against adversarial attacks. In: Leal-Taixé, L., Roth, S. (eds.) Computer Vision – ECCV 2018 Workshops. pp. 267–279. Springer International Publishing, Cham (2019)

29. Tolpin, D.: Population anomaly detection through deep gaussianization. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 1330–1336. SAC '19, ACM, New York, NY, USA (2019). <https://doi.org/10.1145/3297280.3297414>, <http://doi.acm.org/10.1145/3297280.3297414>
30. Xiao, H., Xiao, H., Eckert, C.: Adversarial label flips attack on support vector machines. In: Proceedings of the 20th European Conference on Artificial Intelligence. pp. 870–875. ECAI'12, IOS Press, Amsterdam, NL (2012)
31. Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., Roli, F.: Is feature selection secure against training data poisoning? In: International Conference on Machine Learning. pp. 1689–1698 (2015)
32. Xu, W., Evans, D., Qi, Y.: Feature squeezing: Detecting adversarial examples in deep neural networks. In: Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS) (2017)

## A Mann-Whitney's $U$ -Test

Let  $\mathfrak{D}$  be an arbitrary distribution.

Mann-Whitney's  $U$ -test is a non-parametric hypothesis test. The test assumes that the two compared sample sets  $X_0, X_1$  are independent and that a total order exists on their elements (which is the case for real-valued data such as ML feature vectors).

Assuming that  $X_0 \in_R \mathfrak{D}$ :

- The null hypothesis  $H_0$  is that  $X_1 \in_R \mathfrak{D}$ .
- The alternative hypothesis  $H_1$  is that  $X_1 \in_R \mathfrak{D}'$  for  $\mathfrak{D} \neq \mathfrak{D}'$ .

The test is consistent<sup>6</sup> when, under  $H_1$ ,  $P(X_0 > X_1) \neq P(X_1 > X_0)$ .

The test computes a statistic called  $U$ , which distribution under  $H_0$  is known. When  $\#X_0$  and  $\#X_1$  are large enough, the distribution of  $U$  under  $H_0$  is approximated by a normal distribution of known mean and variance.

$U$  is computed as follows:

1. Merge the elements of  $X_0$  and  $X_1$ . Sort the resulting list by ascending order.
2. Assign a rank to each element of the merged list. Equal elements get as rank the midpoint of their adjusted rankings<sup>7</sup>.
3. Sum the ranks for each set. Let  $R_i$  be this sum for  $X_i$ . Note that if  $n_i = \#X_i$  then  $R_{1-i} = n(n+1)/2 - R_i$ , with  $n = n_0 + n_1$ .
4. Let  $U_i = R_i - \frac{n_i(n_i+1)}{2}$  and  $U = \min(U_0, U_1)$ .

When the  $\#X_i$  are large enough ( $> 20$  elements)  $U$  approximately follows a normal distribution.

Hence, one can check if the value

$$z = \frac{U - \lambda_U}{\sigma_U}$$

follows a standard normal distribution under  $H_0$ , with  $\lambda_U$  being the mean of  $U$ , and  $\sigma_U$  its standard deviation under  $H_0$ :

$$\lambda_U = \frac{n_0 n_1}{2} \quad \text{and} \quad \sigma_U = \sqrt{\frac{n_0 n_1 (n+1)}{12}}$$

However, the previous formulae are only valid when there are no tied ranks. For tied ranks, the following formula is to be used:

$$\sigma_U = \sqrt{\frac{n_0 n_1}{12} \left( (n+1) - \sum_{i=1}^k \frac{t_i^3 - t_i}{n(n-1)} \right)}$$

Because under  $H_0$ ,  $z$  follows a normal distribution, we can estimate the likelihood that the observed values comes from a standard normal distribution, hence getting a related  $p$ -value from the standard normal table.

<sup>6</sup> i.e., its power increases with  $\#X_0$  and  $\#X_1$ .

<sup>7</sup> e.g., in the list 1, 4, 4, 4, 4, 6, the fours all get the rank 3.5

## B Stouffer's $p$ -Value Aggregation Method

$p$  values can be aggregated in different ways [11]. Stouffer [27] observes that the  $z$ -value defined by  $z = \Phi^{-1}(p)$  is a standard normal variable under  $H_0$  where  $\Phi$  is the standard normal CDF. Hence when  $\{p_1, \dots, p_\ell\}$  are translated into  $\{z_1, \dots, z_\ell\}$ , we get a collection of independent and identically distributed standard normal variables under  $H_0$ . To combine the effect of all tests we sum all the  $z_i$  which follows a normal distribution under  $H_0$  with mean 0 and variance  $\ell$ . The global test statistic

$$Z = \frac{1}{\ell} \sum_{i=1}^{\ell} z(p_i)$$

is hence standard normal under  $H_0$  and can thus be reconverted into a  $p$ -value in the standard normal table.

Note that in theory, combining  $p$ -values using Stouffer's method requires that the tests are independent. Other methods can be used for combining  $p$ -values from non-independent tests, e.g. [16,6]. However, these calculations imply that the underlying joint distribution is known, and the derivation of the combination statistics percentiles requires a numerical approximation.