Perceptrons Under Verifiable Random Data Corruption

Jose E. Aguilar Escamilla and Dimitrios I. Diochnos

The University of Oklahoma

Repo: aguilarjose11/Perceptron-Corruption

Perspectives on Robustness

Robustness studied through:

1. **Stability & Reproducibility**
2. **Noise**
3. Poisoning Attacks
4. Missing Data

We focus on **Verifiable Data Corruption** which is related to:

- *Stability and noise*

Verifiable Random Corruption

- Some data points become *corrupted* and can be *verified*.
  - $\mathcal{T} = ((x_1, y_1, f_m), ..., (x_m, y_m, f_m)) \;\; s.t. \;\; f_i \in \{\times, \checkmark\}$
  - Corruption may be noise, attack, repetition, etc.
- Adversary chooses points *randomly*.
- Data undergoes sanitization, **decreasing data size**

How does a classifier perform under such circumstances?

Relation to Influence Functions

Verifiable Data Corruption is related to Influence functions [5].

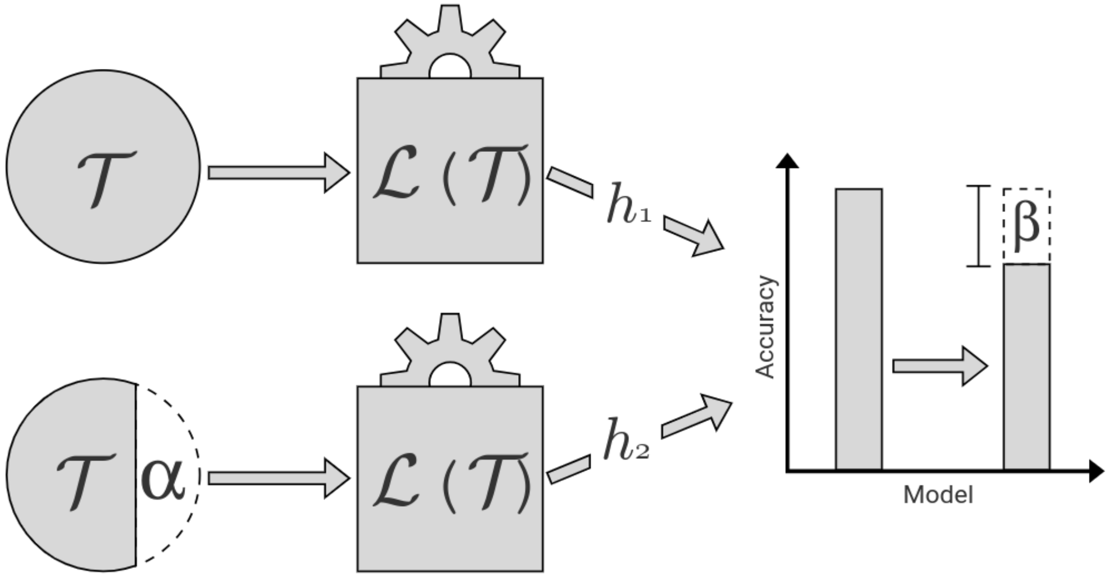
- Adversary has small budget and removes most influential data.

In **Verifiable Data Corruption**, adversary has large budget, and can remove *large amounts* of data at random.

Measuring Stability

(α,β)-stability: Characterize robustness as maximum expected drop in performance.

- No drop larger than β when loosing up to α data.



The UNIVERSITY of OKLAHOMA

The Perceptron

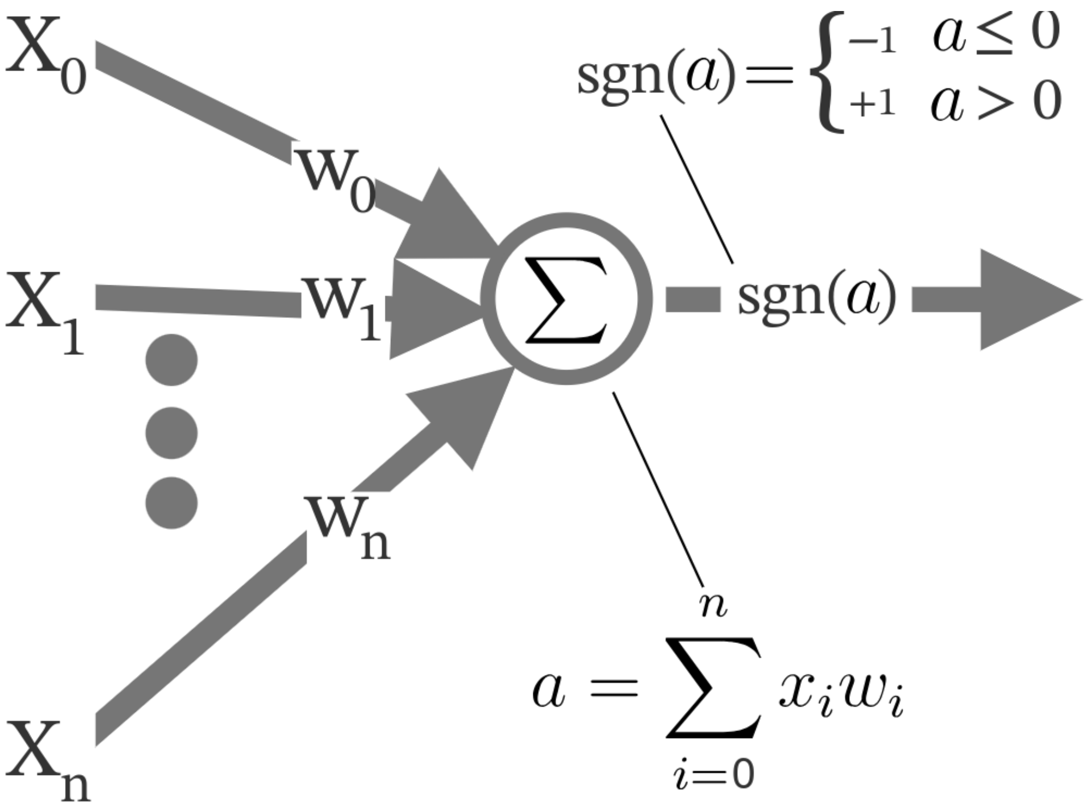
Update rule:

- $\mathbf{w} \leftarrow \mathbf{w} + \eta(y - \ell)x$

Only for linearly-separable data

Pocket Algorithm [1]

- Keep track of accuracy and save best model thus far.



The UNIVERSITY of OKLAHOMA

Corruption Experiment

Simulate corruption by decreasing dataset size [2]

- Split data into buckets/packets and train with decreasing subset size.

Find α and β empirically using collected accuracy

To what extent are perceptrons tolerant to verifiable data corruption?

Datasets

Real

- Datasets from UCI Database.

Dataset	n	Number of Examples	Minority-Class Percentage	Linearly Separable
Iris	4	150	33.3%	yes
Skin	3	245,057	26.0%	no
SPECT	22	267	20.6%	no
Spam	57	4,601	39.4%	no
Bank	95	6,819	3.2%	no

Synthetic

- Linearly-Separable data from randomly-chosen perceptron.
- Non-Linearly-Separable data from n^{th}-degree polynomial with random constants.

Results: Real Data

Real

Average-case comparison:

- (0.25, 0.019)-stable
- **SPECT** is (0.5, 0.035)-stable

Worst-case comparison:

- (0.25, 0.013)-stable

(a) Mean accuracy on real data.

Corruption Level	Iris	Skin	**SPECT**	Spam	Bank
0%	0.999	0.935	**0.753**	0.901	0.966
25%	0.998	0.934	**0.734**	0.898	0.967
50%	0.998	0.933	**0.718**	0.891	0.966
75%	0.998	0.929	**0.709**	0.879	0.968
90%	0.986	0.925	**0.707**	0.856	0.969
95%	0.956	0.922	**0.705**	0.824	0.971
99%	0.727	0.888	**0.715**	0.705	0.986

(b) Worst-case accuracy on real data.

Corruption Level	Iris	Skin	**SPECT**	Spam	Bank
0%	0.966	0.905	**0.643**	0.864	0.961
25%	0.966	0.905	**0.630**	0.864	0.959
50%	0.966	0.910	**0.575**	0.841	0.959
75%	0.966	0.906	**0.602**	0.841	0.953
90%	0.666	0.901	**0.493**	0.766	0.941
95%	0.633	0.878	**0.410**	0.753	0.931
99%	0.333	0.726	**0.205**	0.549	0.878

Results: Real Data with SMOTE

SMOTE

Used Synthetic Minority Over-sampling Technique (SMOTE) [3] to balance datasets.

- (0.5, 0.014)-stable

Table 4: Mean accuracy on real-world data after applying SMOTE on the training set.

Corruption Level	Data Source				
	Iris	Skin	SPECT	Spam	Bank
0%	0.998	0.937	0.703	0.901	0.617
25%	0.999	0.936	0.689	0.899	0.617
50%	0.997	0.935	0.689	0.893	0.616
75%	0.997	0.932	0.654	0.882	0.618
90%	0.996	0.928	0.623	0.858	0.626
95%	0.981	0.925	0.593	0.837	0.637
99%	0.730	0.914	0.534	0.723	0.712

Results: Synthetic Data

Synthetic

Worst-case linearly-separable comparison:

- (0.5, 0.035)-stable

Worst-case non linearly-separable comparison:

- (0.75, 0.035)-stable

(a) Synthetic linearly separable data. Worst-case accuracy shown over 100 runs.

Corruption Level	Data Dimensionality (n) 4	10	25	50	**100**
0%	0.993	0.988	0.980	0.968	**0.961**
25%	0.990	0.985	0.978	0.968	**0.958**
50%	0.991	0.983	0.970	0.961	**0.936**
75%	0.983	0.973	0.961	0.925	**0.885**
90%	0.958	0.951	0.911	0.850	**0.790**
95%	0.948	0.886	0.831	0.753	**0.681**
99%	0.746	0.705	0.623	0.586	**0.555**

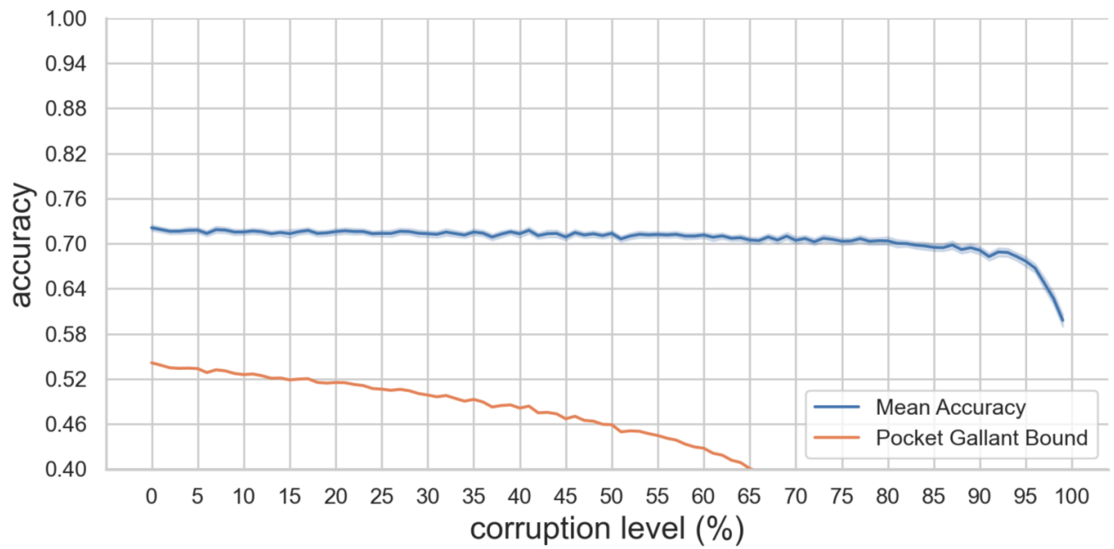
(b) Synthetic non-linearly separable data. Worst-case accuracy shown over 100 runs.

Corruption Level	Data Dimensionality (n) 4	10	25	**50**	100
0%	0.935	0.800	0.676	**0.583**	0.513
25%	0.935	0.801	0.665	**0.591**	0.521
50%	0.921	0.770	0.663	**0.581**	0.536
75%	0.905	0.776	0.643	**0.551**	0.518
90%	0.891	0.730	0.621	**0.558**	0.488
95%	0.863	0.706	0.538	**0.508**	0.478
99%	0.721	0.468	0.491	**0.468**	0.458

Theory Bound vs Stability

Pocket Algorithm has worst-case accuracy bounds.

- The bounds are conservative and lower than actually seen.



The UNIVERSITY of OKLAHOMA

Conclusions and Future Work

Perceptrons are remarkably stable with **good performance** despite large data loss (even in challenging datasets such as SPECT)

Future Work: Look into regularization techniques to apply to pocket algorithm and observe their effect on stability and interpretability. Also, how far can reproducible algorithms [4] take us in this framework?

Thank you

The UNIVERSITY of OKLAHOMA

Works Cited

[1] Gallant, S.I.: Perceptron-based learning algorithms. IEEE Trans. Neural Networks 1(2), 179–191 (1990)

[2] Flansburg, C., Diochnos, D.I.: Wind Prediction under Random Data Corruption (Student Abstract). In: AAAI 2022. pp. 12945 12946. AAAI Press (2022)

[3] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res. 16, 321–357 (2002)

[4] Russell Impagliazzo, Rex Lei, Toniann Pitassi, Jessica Sorrell: Reproducibility in learning. STOC 2022: 818-831

[5] Goldblum, M., Tsipras, D., Xie, C., Chen, X., Schwarzschild, A., Song, D., Madry, A., Li, B., Goldstein, T.: Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses. IEEE Trans. Pattern Anal. Mach. Intell. 45(2), 1563–1580 (2023)

Appendix A: Pocket Algorithm

Algorithm 1: "Pocket" Version of the Perceptron Learning Algorithm

Data: Training examples $\mathcal{T}$.

Result: Best weight vector $\boldsymbol{w}$, in the sense that the induced halfspace classifies $\mathcal{T}$ with as few misclassifications as possible.

$\boldsymbol{\pi} \leftarrow \mathbf{0}$; `/* Initialize to zero all coordinates */`

$\mathrm{run}_{\boldsymbol{\pi}}, \mathrm{run}_{\boldsymbol{w}}, \mathrm{num\_ok}_{\boldsymbol{\pi}}, \mathrm{num\_ok}_{\boldsymbol{w}} \leftarrow \mathbf{0}$;

Randomly pick a training example $(\boldsymbol{x_i}, y_i)$;

if $\boldsymbol{\pi}$ *correctly classifies* $(\boldsymbol{x_i}, y_i)$ **then**

    $\mathrm{run}_{\boldsymbol{\pi}} \leftarrow \mathrm{run}_{\boldsymbol{\pi}} + 1$;

    if $run_{\boldsymbol{\pi}} > run_{\boldsymbol{w}}$ **then**

        Compute $\mathrm{num\_ok}_{\boldsymbol{\pi}}$ by checking every training example;

        if $num\_ok_{\boldsymbol{\pi}} > num\_ok_{\boldsymbol{w}}$ **then**

            $\boldsymbol{w} \leftarrow \boldsymbol{\pi}$; `/* Update the best weight vector found so far */`

            $\mathrm{run}_{\boldsymbol{w}} \leftarrow \mathrm{run}_{\boldsymbol{\pi}}$;

            $\mathrm{num\_ok}_{\boldsymbol{w}} \leftarrow \mathrm{num\_ok}_{\boldsymbol{\pi}}$;

            if *all training examples correctly classified* **then**

                stop (the training examples are separable)

else

    $\boldsymbol{\pi} \leftarrow \boldsymbol{\pi} + y_i \cdot \boldsymbol{x_i}$; `/* Form a new vector of perceptron weights */`

    $\mathrm{run}_{\boldsymbol{\pi}} \leftarrow 0$;

Appendix B: Corruption Experiment

Algorithm 2: Random Data Corruption, Sanitization, and Learning

Data: Training examples $\mathcal{T}$, validation examples Γ.

for R *runs* **do**
    Split $\mathcal{T}$ into B buckets
    for $b = B$ *down to 1* **do**
        Select a random permutation of b buckets to form an uncorrupted set of training examples $\mathcal{T}_{\text{clean}}$ and ignore the examples in $\mathcal{T} \setminus \mathcal{T}_{\text{clean}}$ which are assumed to be verifiably corrupted and thus discardedq m
        Train a perceptron h with the "pocket" algorithm using $\mathcal{T}_{\text{clean}}$ (Algorithm 1)
        Collect the accuracy of h over the validation examples Γ

Appendix C: Data Statistics

Dataset	n	Number of Examples	Minority-Class Percentage	Linearly Separable
Iris	4	150	33.3%	yes
Skin	3	245,057	26.0%	no
SPECT	22	267	20.6%	no
Spam	57	4,601	39.4%	no
Bank	95	6,819	3.2%	no

Appendix D: ¿Why the Perceptron?

1. Simple
2. Explainable
   - Recent push for explainability in policy (e.g., EU's GDPR)
3. Theoretically well studied
   - Bounds on required data to reach certain accuracy with high probability