



# Perceptrons Under Verifiable Random Data Corruption

Jose E. Aguilar Escamilla   
jose.efraim.a.e@gmail.com

Dimitrios I. Diochnos   
diochnos@ou.edu

University of Oklahoma

## Abstract

We study perceptrons when datasets are randomly corrupted by noise and subsequently such corrupted examples are discarded from the training process. Overall, perceptrons appear to be remarkably stable; their accuracy drops slightly when large portions of the original datasets have been excluded from training as a response to verifiable random data corruption. Furthermore, we identify a real-world dataset where it appears to be the case that perceptrons require longer time for training, both in the general case, as well as in the framework that we consider. Finally, we explore empirically a bound on the learning rate of Gallant’s “pocket” algorithm for learning perceptrons and observe that the bound is tighter for non-linearly separable datasets.

**Keywords:** Perceptrons, Stability, Verifiable Random Data Corruption, Reduced Sample Size

## 1 Introduction

The advances in machine learning during the last few years are without a precedent with ChatGPT [4] being perhaps the most remarkable example of the tremendous pace by which these advances are made. Nevertheless, there are concerns about the future of artificial intelligence and machine learning, the extent to which we can control the advances in these disciplines, and ultimately the impact that such advances will have in our daily lives [22].

Supervised machine learning is largely developed around the idea of *probably approximately correct learning* [28], which aims to generate, with high probability, a model that is highly accurate with respect to some ground truth function that is being learnt. For this reason, for the majority of applications, perhaps the most important metric that one cares on optimizing, is that of *accuracy* (or its dual, *error rate*), on a *validation set*. Nevertheless, we are also interested in other aspects of the learnt models, that we typically put under the broader umbrella of *trustworthy machine learning* [29]; the main axes being, *robustness* to noise [21] and adversaries [30], *interpretability* [23], and *fairness* [1], but one may also be interested in other related topics; e.g., *distribution shift* [5], and others.

Hence, within trustworthy machine learning, and especially because of the need of explainable predictions, mechanisms that are inherently interpretable are put further into scrutiny, or attempts are made so that one can substitute components of black-box methods that are difficult to explain with such mechanisms, while still maintaining good predictive accuracy [25]. In this context, a basic, but popular approach, is the use of *linear models* for classification and it is such models that we explore in this work; namely, *perceptrons*.

## 1.1 Related Work

Different frameworks have been developed in order to study the robustness of classifiers and how one can attack or defend the classifiers in different contexts.

**Stability and Reproducibility.** Algorithms that do not overfit are *replace-one-stable* [27]; that is, replacing a particular example  $(\mathbf{x}_0, y_0)$  in the training set, will yield a model that predicts a label on  $\mathbf{x}_0$  *similar* to  $y_0$ , had the learning algorithm used the original training set which included the example  $(\mathbf{x}_0, y_0)$ ; see, e.g., [27]. Other notions of stability are also important and are studied in practice; e.g., how random seeds affect the performance of a learnt model [9]. Moreover, a recent line of work explores *reproducible algorithms* [16], where the idea is that the learning algorithm will generate *exactly the same model* when fed with two different samples drawn from the *same distribution*.

**Noise.** Beyond stability and reproducibility, data collected can be noisy, and a rich body of literature has been developed in order to understand better what is and what is not possible when datasets are corrupted by noise. A notable noise model is *malicious noise*, where both instances and labels can be corrupted arbitrarily by some adversary. In this model, it can be shown [17] that if we want to allow the possibility of generating a model that has error rate less than  $\varepsilon$ , then, the original dataset cannot suffer malicious corruption at a rate larger than  $\frac{\varepsilon}{1+\varepsilon}$ . In other words, even few examples that are carefully crafted and included in the learning process are enough to rule out models with low error rates.

**Poisoning Attacks.** While noise models typically assume the corruption of the examples in an online manner, *poisoning attacks* [3] typically consider adversaries who have access to the entire training set and decide how to inject or modify examples based on the full information of the dataset. Different models and defense mechanisms can be devised against poisoned data [13].

One approach for poisoning, corresponds to *clean-label attacks* [26], where the adversary may inject malicious examples in the training set, but their labels have to respect the ground truth of the function that is being learnt. In other words, the idea is to change the original distribution of instances to a more malicious one and make learning harder; difficult distributions have also been studied even for perceptrons, where it can be shown that the perceptrons may converge to solutions with significantly different time-complexity rates depending on the distribution [2]. A different line of work studies the *influence* that individual training examples have on the predictions of the learnt model, by being present or absent during learning [18]. Hence, instead of introducing misinformation in the dataset, one tries to understand how important different examples are for the learning process. In other words, how would the model’s prediction change if we did not have access to a particular training example? Of course, one can introduce misinformation into the dataset; e.g., [19] considers the problem of designing imperceptible perturbations on training examples that can change the predictions of the learnt model. Overall, poisoning attacks study the brittleness of learning algorithms when training happens in the presence of adversaries.

As a last remark, one can also consider adversaries after training has been completed. In this direction, we find *evasion attacks*, also known as *adversarial examples* [14]. We note,

however, that such adversarial settings are outside of the scope of the current work, as we deal with *training-time* attacks.

**Missing Data.** Concluding our brief literature review, an aspect that is quite orthogonal to the approaches discussed above, is that of dealing with missing data in our datasets. The reasons for the emergence of such datasets can range anywhere from pure chance (e.g., a questionnaire of a study subject is accidentally lost), to strategic information hiding (e.g., university applicants reveal favorable scores and hide potentially unfavorable ones in their applications) [20]. Typically, entries with missing data will either be completely ignored from the dataset and the training process, or some method will be devised in order to fill-in the missing entries based on the information of the rest of the entries in the dataset [12] so that they can subsequently be used for training. Perhaps the main difference with noise and poisoning attacks is the fact that the learner knows that certain entries are completely unknown in the input. Finally, we note that dealing with missing entries may also happen during evasion attacks [7], but we stress again that evasion attacks are outside of the scope of our work.

## 1.2 Our Setting: Verifiable Random Data Corruption

We explore the robustness of linear models used for classification. In particular, we study the perceptron learning algorithm [24] when the data may, or may not, be linearly separable; we use Gallant’s “pocket” algorithm [11] which can deal with non-linearly separable data. Regarding the adversarial setting, we consider situations where the training data has been randomly corrupted by noise (e.g., transmission over an unreliable medium). However, we further assume, as is usually the case in practice, that there are flags (e.g., checksums) that indicate whether or not the information content of individual examples has been modified. In this context, we explore the following question:

*To what extent are perceptrons tolerant of verifiable random data corruption?*

One natural way of dealing with such a scenario is to simply neglect the verifiably corrupted data altogether from the learning process. Hence, we are interested in understanding the behavior of perceptrons, in the *average case* as well as in a *worst case* sense, under such a simple *sanitization method* that effectively reduces the sample size used for learning. This adversarial setup and the sanitization approach has been considered before in a regression setting [10].

However, there are close connections to other adversarial models. For example, within clean-label attacks, the adversary may duplicate examples that already appear in the dataset. While such an approach may affect dramatically algorithms that rely on statistical properties of the data (e.g., decision trees), using perceptrons is very close to our context. The reason is that the weights of the learnt perceptron are a linear combination of the misclassified examples and hence the same solution can be obtained even when duplicate examples are omitted. Compared to the work of influence functions [18], our work explores the possibility of arbitrarily large amounts of data being removed from the training process, instead of carefully removing few but the most influential ones.

As a summary, we explore a mellow corruption scheme which has the benefits of (i) being easier to analyze, and (ii) occurring quite naturally in practice.

## 2 Preliminaries and Background

**Notation.** We study *binary classification* and we use  $\mathcal{Y} = \{-1, +1\}$  to denote the set of *labels*, where  $-1$  (resp.  $+1$ ) corresponds to the *negative class* (resp. *positive class*). We use  $\mathcal{X}$  to denote the set of *instances*, corresponding to real vectors; i.e.,  $\mathcal{X} = \mathbb{R}^n$ . Note that throughout the paper we use  $n$  to denote the dimension of an instance. For two vectors  $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ , we denote their inner product  $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^n a_i b_i$ . Typically, the learner has access to a collection  $\mathcal{T} = ((\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m))$  of *training examples* that are drawn iid from a distribution  $\mathcal{D}$  governing  $\mathcal{X} \times \mathcal{Y}$ . Hence, learning is about selecting one appropriate *model*  $h: \mathcal{X} \rightarrow \mathcal{Y}$  from the *model space*  $\mathcal{H}$ , based on  $\mathcal{T}$ . That is, for a learning algorithm  $\mathcal{L}$ , we have  $\mathcal{L}(\mathcal{T}) = h$ . We also use the function which returns the sign of its argument; i.e.,  $\text{sgn}(z) = +1$  if  $z > 0$  and  $\text{sgn}(z) = -1$  if  $z \leq 0$ . We consider the “*pocket*” variant of the perceptron, which is suitable also for non-linearly separable data. Furthermore, we need the following definition.

**Definition 1** ( $(\alpha, \beta)$ -Stability Against Random Data Corruption). *Let  $\alpha, \beta \in [0, 1]$ . Let  $\mathcal{L}$  be a learning algorithm, let  $\mathcal{T}$  be a dataset, and let  $\mathcal{L}(\mathcal{T}) = h$ . Moreover, let  $A_h$  be the accuracy of the learnt model  $h \in \mathcal{H}$  on  $\mathcal{T}$ . We say that  $\mathcal{L}$  is  $(\alpha, \beta)$ -stable on  $\mathcal{T}$ , if when one removes up to an  $\alpha$ -fraction of  $\mathcal{T}$  chosen uniformly at random, and then use  $\mathcal{L}$  to learn a model  $h'$  using the remaining examples, then it holds that  $A_{h'} \geq A_h - \beta$ .*

The fine point is that initially our learner has access to examples of the form  $\mathcal{T}_v = ((x_1, y_1, f_1), \dots, (x_m, y_m, f_m))$ , where  $f_i \in \{\blacktimes, \blacktriangleright\}$ , indicating whether the information content of  $x_i$  and  $y_i$  has been modified or not. As a consequence, the dataset  $\mathcal{T}_v$  is sanitized by simply dropping all the examples of the form  $(x_i, y_i, \blacktimes)$  and learning with the rest. See also Section 3.1 for more details.

### 2.1 The Perceptron Learning Algorithm

A perceptron  $h$  maintains a set of weights  $\mathbf{w} \in \mathbb{R}^n$  and classifies an instance  $\mathbf{x}$  with label  $\ell$  according to the rule  $h(\mathbf{x}) = \ell = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle)$ . Upon predicting  $\ell$ , the perceptron updates its weight vector  $\mathbf{w}$  using the rule  $\mathbf{w} \leftarrow \mathbf{w} + \eta(y - \ell)\mathbf{x}$ , where  $\eta$  is the *learning rate* and  $y$  is the correct label. This rule, when applied on the misclassified examples (weights are updated only when mistakes occur), learns a *halfspace* that correctly classifies all training data assuming linear separability.

### 2.2 The Pocket Algorithm

The pocket algorithm is more useful as it applies the same update rule as the perceptron, but also stores (in its *pocket*) the best weights that have been discovered during the learning process; thus, allowing us to identify good solutions even for non-linearly separable data. Algorithm 1 has the details, where by following Gallant’s notation [11], we have the following variables:  $\boldsymbol{\pi}$  are the current weights of the perceptron;  $\mathbf{w}$  are the weights of the best perceptron that we have encountered so far;  $\text{run}_{\boldsymbol{\pi}}$  corresponds to the number of consecutive correct classifications using the current weights, when we select examples from  $\mathcal{T}$  at random;  $\text{run}_{\mathbf{w}}$  corresponds to the number of consecutive correct classifications that the weights that we have in our pocket (i.e., the best weights that we have encountered so far), when we select examples from  $\mathcal{T}$  at random;  $\text{num\_ok}_{\boldsymbol{\pi}}$  is the true number of examples from  $\mathcal{T}$  that the

---

**Algorithm 1:** “Pocket” Version of the Perceptron Learning Algorithm

---

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

```
1  $\boldsymbol{\pi} \leftarrow \mathbf{0}$ ; /* Initialize to zero all coordinates */
2  $\text{run}_{\boldsymbol{\pi}}, \text{run}_{\mathbf{w}}, \text{num\_ok}_{\boldsymbol{\pi}}, \text{num\_ok}_{\mathbf{w}} \leftarrow \mathbf{0}$ ;
3 Randomly pick a training example  $(\mathbf{x}_i, y_i)$ ;
4 if  $\boldsymbol{\pi}$  correctly classifies  $(\mathbf{x}_i, y_i)$  then
5    $\text{run}_{\boldsymbol{\pi}} \leftarrow \text{run}_{\boldsymbol{\pi}} + 1$ ;
6   if  $\text{run}_{\boldsymbol{\pi}} > \text{run}_{\mathbf{w}}$  then
7     Compute  $\text{num\_ok}_{\boldsymbol{\pi}}$  by checking every training example;
8     if  $\text{num\_ok}_{\boldsymbol{\pi}} > \text{num\_ok}_{\mathbf{w}}$  then
9        $\mathbf{w} \leftarrow \boldsymbol{\pi}$ ; /* Update the best weight vector found so far */
10       $\text{run}_{\mathbf{w}} \leftarrow \text{run}_{\boldsymbol{\pi}}$ ;
11       $\text{num\_ok}_{\mathbf{w}} \leftarrow \text{num\_ok}_{\boldsymbol{\pi}}$ ;
12      if all training examples correctly classified then
13        stop (the training examples are separable)
14 else
15    $\boldsymbol{\pi} \leftarrow \boldsymbol{\pi} + y_i \cdot \mathbf{x}_i$ ; /* Form a new vector of perceptron weights */
16    $\text{run}_{\boldsymbol{\pi}} \leftarrow 0$ ;
```

---

current weights  $\boldsymbol{\pi}$  of the perceptron classify correctly; and  $\text{num\_ok}_{\mathbf{w}}$  is the true number of examples from  $\mathcal{T}$  that the weights  $\mathbf{w}$  that we have in our pocket classify correctly. Hence, the “best” pocket weights correspond to the weights that classified correctly the maximum number of examples from the collection  $\mathcal{T}$  during training.

Gallant has provided sample complexity bounds for different topologies of perceptron networks. Proposition 1 below is a simplification of a result from [11], adapted to the “single-cell” model (i.e., a single neuron; a perceptron).

**Proposition 1** (See [11]). *Let  $\varepsilon$  be the true error and  $\varepsilon^\circ$  be the error over  $\mathcal{T}$ , after learning. Let  $\mathcal{S} = (\varepsilon - \varepsilon^\circ)/\varepsilon$ ; i.e.,  $\mathcal{S}$  is the slack given to the algorithm between the true error  $\varepsilon$  and empirical error  $\varepsilon^\circ$ . Also, let  $L = \|\mathbf{w}\|_2 = \sqrt{\sum_{j=0}^n w_j^2}$  be the length of the weight vector of the learnt perceptron. Then, with probability at least  $1 - \delta$ , the learnt perceptron will have true error rate at most  $\varepsilon$ , if the number of training examples  $m$  is larger than*

$$\min \left\{ \frac{8}{\mathcal{S}^2 \varepsilon} \max \left\{ \ln(8/\delta), \min\{2(n+1), 4(n+1) \log_2(e)\} \ln\left(\frac{16}{\mathcal{S}^2 \varepsilon}\right) \right\}, \frac{\ln(1/\delta) + (n+1) \ln(2L+1)}{\mathcal{S}^2 \varepsilon} \min\left\{\frac{1}{2\varepsilon}, 2\right\} \right\}.$$

### 3 Methodology and Resources

#### 3.1 Random Data Corruption and Sanitization

We simulate the process of verifiable random data corruption affecting our dataset, by ignoring random groups of examples in our dataset and using the rest (where the checksums indicate no tampering) for training. Algorithm 2 summarizes this approach, where the whole evaluation

---

**Algorithm 2:** Random Data Corruption, Sanitization, and Learning

---

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

```
1 for  $R$  runs do
2   Split  $\mathcal{T}$  into  $B$  buckets
3   for  $b = B$  down to 1 do
4     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 discarded
5     Train a perceptron  $h$  with the ‘‘pocket’’ algorithm using  $\mathcal{T}_{\text{clean}}$  (Algorithm 1)
6     Collect the accuracy of  $h$  over the validation examples  $\Gamma$ 
```

---

process is repeated  $R$  times in order to smooth the results and understand better the expected, as well as the extreme, values of the performance of the learnt model. We test the above method using both synthetic, as well as real-world, datasets, of varying dimensions, in order to better understand the behavior of perceptrons when the available data has been corrupted (verifiably) at random instances.

### 3.2 Description of Synthetic Datasets

We constructed two synthetic datasets by randomly sampling 3,000 points from a specified dimensional space and labeling them using a pre-selected linear or non-linear model. In particular, for each dataset, we sampled  $n \in \{4, 10, 25, 50, 100\}$  dimensional spaces with attribute values ranging between  $-10$  and  $+10$ , following a uniform distribution in that interval. We then added an extra parameter with a constant value of  $+1$  to be used as bias. For the linearly-separable synthetic dataset, we randomly sampled a baseline perceptron’s parameters and we used it to produce the label of each point. For the non-linearly separable case, we used an  $n$ -degree polynomial of the form  $c_0 + c_1x_1 + c_2x_2^2 + c_3x_3^3 + \dots + c_nx_n^n$ , with randomly-chosen constants between  $-1$  and  $+1$ . We obtained the labels by using the sign function of the output from the polynomial.

### 3.3 Description of Real-World Datasets

We also performed experiments with five real-world datasets. Two datasets had comparable dimension; ‘Iris’ is linearly separable while the skin segmentation (‘Skin’) dataset is not. In order to get a better picture we explored the behavior of three more datasets of larger dimension: ‘SPECT’, ‘Spam’, and ‘Bank’, corresponding respectively to the SPECT heart dataset, the spambase dataset, and the Taiwanese bankruptcy prediction dataset. Table 1 has details. All datasets are available at the UCI repository.<sup>1</sup>

Table 1: Summary of real-world datasets used in our experiments.

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

<sup>1</sup>Homepage: <https://archive.ics.uci.edu>

Table 2: Worst-case accuracy of perceptrons on synthetic datasets.

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

Corruption Level	Data Dimensionality ( $n$ )				
	4	10	25	50	100
0%	0.993	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

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

### 3.4 Training Set, Validation Set, Buckets, and Smoothing

Apart from the Iris and SPECT datasets which have only 150 and 267 instances respectively, in the other cases (synthetic, or real-world data), we sampled 3,000 data points at random. Such a number is more than sufficient for low dimensions, but is typically too small for traditional values obtained from statistical learning theory for datasets with  $n \geq 50$  dimensions, when one wants (excess) error rate  $\varepsilon - \varepsilon^\circ \leq 1\%$  with confidence  $1 - \delta \geq 99\%$ . Nevertheless, this amount seems reasonable given the size, in terms of the number of examples, of the real-world datasets that we explore – and in general, this is a reasonable size for real-world datasets. In every case, we performed an 80-20 split for the creation of the training examples  $\mathcal{T}$  and validation examples  $\Gamma$  that are mentioned in Algorithm 2. We then subdivided  $\mathcal{T}$  into  $B = 100$  buckets of equal size and applied Algorithm 2 for the calculation of the performance (on the validation set  $\Gamma$ ) of the models obtained, over  $R = 100$  runs, using the pocket algorithm. Each run would process at most  $3,000|\mathcal{T}|$  training examples.

## 4 Experimental Results and Discussion

Source code is available at [github.com/aguiarjose11/Perceptron-Corruption](https://github.com/aguiarjose11/Perceptron-Corruption).

### 4.1 Experimental Results and Discussion on Synthetic Datasets

Table 2 presents the worst-case accuracy attained by perceptrons when learning synthetic datasets of different dimensions, covering both linearly separable as well as non-linearly separable data. Worst-case accuracy drops more than 3.5% only after 50% of the linearly-separable data has been corrupted, or more than 75% of the non-linearly separable data has been corrupted. In other words, by Definition 1, perceptrons are  $(0.5, 0.035)$ -stable for linearly separable data and  $(0.75, 0.035)$ -stable for non-linearly separable data, in the worst case. Using a similar table (not shown in the paper) for the average-case accuracy, perceptrons were, *on average*,  $(0.5, 0.018)$ -stable on linearly separable data and  $(0.75, 0.018)$ -stable on non-linearly separable data. Another takeaway is that perceptrons are  $(0.25, 0.01)$ -stable in the worst case, regardless of the synthetic dataset.

## 4.2 Experimental Results and Discussion on Real-World Datasets

Table 3 shows the average-case and worst-case performance of perceptrons on the datasets that we described in Section 3.3. Apart from SPECT, perceptrons are  $(0.5, 0.023)$ -stable and

Table 3: Mean and worst-case accuracy attained by perceptrons when learning real-world datasets. Values calculated over 100 runs.

(a) Mean accuracy on real data.						(b) Worst-case accuracy on real data.					
Corruption Level	Data Source					Corruption Level	Data Source				
	Iris	Skin	SPECT	Spam	Bank		Iris	Skin	SPECT	Spam	Bank
0%	0.999	0.935	0.753	0.901	0.966	0%	0.966	0.905	0.643	0.864	0.961
25%	0.998	0.934	0.734	0.898	0.967	25%	0.966	0.905	0.630	0.864	0.959
50%	0.998	0.933	0.718	0.891	0.966	50%	0.966	0.910	0.575	0.841	0.959
75%	0.998	0.929	0.709	0.879	0.968	75%	0.966	0.906	0.602	0.841	0.953
90%	0.986	0.925	0.707	0.856	0.969	90%	0.666	0.901	0.493	0.766	0.941
95%	0.956	0.922	0.705	0.824	0.971	95%	0.633	0.878	0.410	0.753	0.931
99%	0.727	0.888	0.715	0.705	0.986	99%	0.333	0.726	0.205	0.549	0.878

$(0.75, 0.023)$ -stable in the worst case; i.e., perceptrons are similarly stable compared to our observations on the synthetic datasets. However, on SPECT, perceptrons are only  $(0.5, 0.068)$ -stable in the worst case. Finally, perceptrons, including SPECT, are  $(0.25, 0.013)$ -stable in the worst case.

Note that the different classes are represented in an imbalanced way in the datasets; see Table 1 for the ratio of the minority class in every case. Class imbalance can be a nuisance for classification [8, 15]. One trivial solution under class imbalance is to predict according to the majority class. Under no corruption, and with the exception of SPECT, perceptrons were able to form a better solution than this trivial approach, whereas for SPECT it was the case that not even such a good solution was found. Nevertheless, in order to better understand how far one can go with imbalanced datasets, we also applied the Synthetic Minority Over-sampling Technique (SMOTE) [6], a popular method for dealing with class imbalance. The experiments using SMOTE had mixed results: SMOTE helped with the accuracy on the Skin dataset, but hurt the accuracy of the models in the other cases. Also, in some cases we identified a tradeoff between less accuracy and higher stability, but this observation was not consistent among all the datasets that we explored. Table 4 has details for the average case when SMOTE was applied in the training dataset.

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



### 4.3 Empirical Investigation of Gallant’s Bound

In all of our experiments with separable and non-separable data, we compared the empirical accuracy that we obtained in our experiments, with the (*lower bound*) on the *accuracy* that is implied by Proposition 1 when we require failure probability  $\delta = 1\%$ . Proposition 1 was tighter on non-linearly separable data. Figure 1 gives an example of such a comparison on the non-linearly separable synthetic dataset of dimension  $n = 25$ .

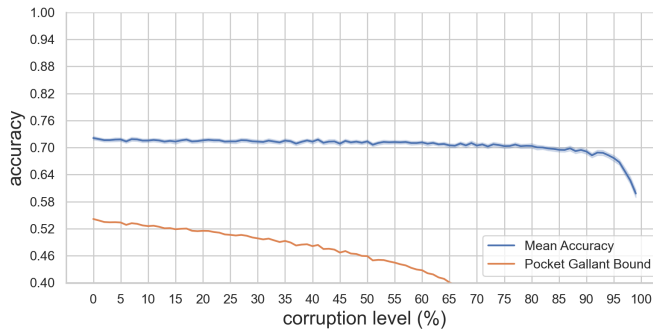


Figure 1: Mean accuracy of perceptrons learnt on synthetic non-linearly separable datasets ( $n = 25$ ) against the lower bound on the accuracy obtained from Proposition 1 using  $\delta = 0.01$ .

## 5 Conclusion

Perceptrons appear to be remarkably stable when learning with reduced sample sizes. In all of our experiments removing up to 25% of the initial training sets would lower the accuracy of the learnt perceptron by not more than 1.3%. Furthermore, SPECT appears to be a harder dataset compared to all the other datasets tested; perceptrons failed to find even trivial solutions such that one can unconditionally predict according to the majority class and achieve better accuracy. This phenomenon complements [2] in the sense that SPECT appears to be a dataset where perceptrons require an extended amount of time for identifying a good solution. Finally, we observed empirically that Gallant’s bound on the pocket algorithm is tighter for non-linearly separable data.

One avenue for future work could be the exploration of penalty mechanisms embedded into the “pocket” algorithm, along the lines of regularization, so that we can obtain even more interpretable solutions for problems of interest, or accelerate learning similar to [10]. Another idea is the investigation of the reproducible weak halfspace learner from [16] and study its stability in this framework.

**Acknowledgements.** Part of the work was performed at the OU Supercomputing Center for Education & Research (OSCER) at the University of Oklahoma. The work was supported by the second author’s startup fund. The first author worked on this topic while he was an undergraduate McNair Scholar.

## References

- [1] Barocas, S., Hardt, M., Narayanan, A.: Fairness and Machine Learning: Limitations and Opportunities. fairmlbook.org (2019), <http://www.fairmlbook.org>
- [2] Baum, E.: The Perceptron Algorithm Is Fast for Non-Malicious Distributions. In: NeurIPS 1989. vol. 2, pp. 676–685. Morgan-Kaufmann (1989)
- [3] Biggio, B., Nelson, B., Laskov, P.: Poisoning Attacks against Support Vector Machines. In: ICML 2012. icml.cc / Omnipress (2012)
- [4] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., et al: Language models are few-shot learners. In: NeurIPS 2020, virtual (2020)
- [5] Quiñonero Candela, J., Sugiyama, M., Schwaighofer, A., Lawrence, N.D.: Dataset Shift in Machine Learning. The MIT Press (2008)
- [6] 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)
- [7] Dekel, O., Shamir, O., Xiao, L.: Learning to classify with missing and corrupted features. Mach. Learn. **81**(2), 149–178 (2010)
- [8] Diochnos, D.I., Trafalis, T.B.: Learning Reliable Rules under Class Imbalance. In: SDM. pp. 28–36. SIAM (2021)
- [9] Fellicious, C., Weißgerber, T., Granitzer, M.: Effects of Random Seeds on the Accuracy of Convolutional Neural Networks. In: LOD 2020, Revised Selected Papers, Part II. Lecture Notes in Computer Science, vol. 12566, pp. 93–102. Springer (2020)
- [10] Flansburg, C., Diochnos, D.I.: Wind Prediction under Random Data Corruption (Student Abstract). In: AAI 2022. pp. 12945–12946. AAI Press (2022)
- [11] Gallant, S.I.: Perceptron-based learning algorithms. IEEE Trans. Neural Networks **1**(2), 179–191 (1990)
- [12] García-Laencina, P.J., Sancho-Gómez, J., Figueiras-Vidal, A.R.: Pattern classification with missing data: a review. Neural Comput. Appl. **19**(2), 263–282 (2010)
- [13] 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)
- [14] Goodfellow, I.J., McDaniel, P.D., Papernot, N.: Making machine learning robust against adversarial inputs. Commun. ACM **61**(7), 56–66 (2018)
- [15] He, H., Garcia, E.A.: Learning from Imbalanced Data. IEEE Transactions on Knowledge and Data Engineering **21**(9), 1263–1284 (2009)
- [16] Impagliazzo, R., Lei, R., Pitassi, T., Sorrell, J.: Reproducibility in learning. In: STOC '22. pp. 818–831. ACM (2022)

- [17] Kearns, M.J., Li, M.: Learning in the Presence of Malicious Errors. *SIAM J. Comput.* **22**(4), 807–837 (1993)
- [18] Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. In: *ICML 2017. Proc. of Mach. Learn. Res.*, vol. 70, pp. 1885–1894. PMLR (2017)
- [19] Koh, P.W., Steinhardt, J., Liang, P.: Stronger data poisoning attacks break data sanitization defenses. *Mach. Learn.* **111**(1), 1–47 (2022)
- [20] Krishnaswamy, A.K., Li, H., Rein, D., Zhang, H., Conitzer, V.: Classification with Strategically Withheld Data. In: *AAAI 2021*. pp. 5514–5522. AAAI Press (2021)
- [21] Laird, P.D.: *Learning from Good and Bad Data*, vol. 47. Springer Science & Business Media (2012)
- [22] Marcus, G.: Hoping for the Best as AI Evolves. *Commun. ACM* **66**(4), 6–7 (Mar 2023), <https://doi.org/10.1145/3583078>
- [23] Molnar, C.: *Interpretable Machine Learning*. Independently Published, 2 edn. (2022), <https://christophm.github.io/interpretable-ml-book>
- [24] Rosenblatt, F.: *Principles of Neurodynamics*. Spartan Books, New York (1962)
- [25] Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* **1**(5), 206–215 (2019)
- [26] Shafahi, A., Huang, W.R., Najibi, M., Suci, O., Studer, C., Dumitras, T., Goldstein, T.: Poison frogs! targeted clean-label poisoning attacks on neural networks. In: *NeurIPS 2018*. pp. 6106–6116 (2018)
- [27] Shalev-Shwartz, S., Ben-David, S.: *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press (2014)
- [28] Valiant, L.G.: A theory of the learnable. *Commun. ACM* **27**(11), 1134–1142 (1984)
- [29] Varshney, K.R.: *Trustworthy Machine Learning*. Independently Published, Chappaqua, NY, USA (2022)
- [30] Vorobeychik, Y., Kantarcioglu, M.: *Adversarial Machine Learning*. Synthesis lectures on artificial intelligence and machine learning, # 38, Morgan & Claypool, San Rafael, California (2018)