Universiteit Leiden
The Netherlands

# Robust online learning

Tiedemann, F.

**Citation**
Tiedemann, F. (2020). *Robust online learning*.

| | |
|---|---|
| Version: | Not Applicable (or Unknown) |
| License: | License to inclusion and publication of a Bachelor or Master thesis in the Leiden University Student Repository |
| Downloaded from: | https://hdl.handle.net/1887/3596180 |

**Note:** To cite this publication please use the final published version (if applicable).

# Tiedemann Thesis

*door* Frederik Tiedemann

**Datum van inzending:** 21-nov.-2019 05:14PM (UTC+0100)
**Inzending-ID:** 1218732443
**Bestandsnaam:** Thesis_Draft.pdf (850.68K)
**Aantal woorden:** 12500
**Aantal tekens:** 58740

# Robust Online Learning

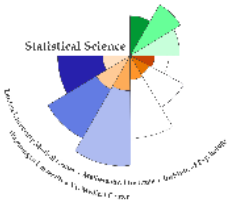
Frederik Tiedemann (s1514210)

Thesis advisor: Dr. Tim van Erven

MASTER THESIS

Defended on Month Day, 2019

Specialization: Data Science


**STATISTICAL SCIENCE**
**FOR THE LIFE AND BEHAVIOURAL SCIENCES**

**Abstract**

Online learning methods process data sequentially. [10] They are known for their computational efficiency when dealing with large data sets and are one of the fundamental tools used in machine learning, e.g. for parameter optimization. Most methods such as online gradient descent update parameters based on the gradients of a loss function evaluated at the data point that is currently being processed.

While existing methods are robust in the sense that they do not rely on probabilistic assumptions and can handle adversarial learning, they are not robust to outliers that cause large prediction errors. These outliers manifest themselves as large gradients that cause large updates of the target parameters.

Existing ad hoc solutions to deal with large gradients are to either clip the gradients at a user-specified level or to ignore gradients that exceed this level. There is, however, no corresponding theory and it is very difficult to specify the correct level manually: If it is set too large, no outliers are detected, and if it is set too small, many data points will be incorrectly treated as outliers.

This thesis introduces relevant data contamination models and reviews literature on robust mean estimation and robust stochastic optimization. Based on this, we propose two contamination models for the online learning framework. We further propose robust regret, a measure of how well online optimization algorithms learn from contaminated data. We then show that by using adaptive algorithms, i.e. algorithms that cope well with changes in the underlying data structure, we can achieve low robust regret in settings with adversarial outliers. We discuss two specific algorithms proposed in [12] and [4] and analyze how many outliers each can handle. Lastly, we show that while both algorithms' analyses depend on the boundedness of the loss function, they in fact only require the loss on the *good* data points to be bounded.

We test this via practical implementations, comparing standard online learning algorithms (base learners) to their robust counterparts using simulated data in the regression setting.

# Contents

# 1 Introduction

The problem of robust estimation has been studied in statistics for over fifty years. Here, we first define the settings that we consider with respect to robustness throughout the remainder of the thesis: regression, mean estimation, stochastic optimization and adversarial online learning.

We then provide a brief review of robust statistics and its applications to estimation, as well as approaches to robustness stemming from machine learning. Lastly, we discuss their application to online learning.

We first define the settings when *not* taking into account robustness.

**Definition 1.1.** (Regression setting)
Data points $\{y_i, \mathbf{x}_i\}^n \in \mathbb{R}^d$ are sampled i.i.d. from some distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, i.e. $\mathcal{D}$ is a joint distribution over the domain sets of our features and response variable. The goal is then to estimate a parameter vector $\mathbf{h}$ such that the error is minimized. The error is commonly measured via the $\ell_2$ norm of the difference between the estimated parameter vector and the true parameter vector.
**Definition 1.2.** (Mean estimation setting)
Data points $\{\mathbf{x}_i\}^n \in \mathbb{R}^d$ are sampled i.i.d. from some distribution $\mathcal{D}$. The goal is then to estimate the true mean of $\mathcal{D}$, $\mu_{\mathcal{D}}$, such that the estimate is close to the true mean.
**Definition 1.3.** (Stochastic optimization setting)
Data points $\{(\mathbf{x}_i, y_i)\}^n$ are sampled i.i.d. from some distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, with the goal of finding a parameter vector $\mathbf{h} \in \mathcal{H}, \mathcal{H} \subseteq \mathbb{R}^d$, where $\mathcal{H}$ denotes the parameter space, that minimizes the risk function $\mathbb{E}[\mathcal{L}(\mathbf{h}, \mathbf{x}, y)]$, where $\mathcal{L}$ is some loss function convex in $\mathbf{h}$.
**Definition 1.4.** (Adversarial Online Learning) Data $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathcal{X}$ is received over time and we make no further assumptions regarding its origins. At every time point $t$ the online player makes a decision $h \in \mathcal{H}$, where $\mathcal{H}$ is a bounded convex set denoting the decision space. Then he observes an entry $\mathbf{x}_t$ from the data stream and nature chooses a convex loss function $f_t : \mathcal{H} \mapsto \mathbb{R}$.

Instead of finding parameter estimates s.t. the difference between the estimate and the true parameter vector is small in some norm, as in e.g. the stochastic optimization setting, the performance of online learning algorithms is measured in terms of regret: the difference between the total loss incurred and the best fixed decision a player could have taken in hindsight. [20] Adversarial online learning will be introduced more formally in chapter 4.

The regression and mean estimation settings are commonly used in statistics. In machine learning research, especially mean estimation has become increasingly important given its potential applications to robust stochastic optimization.

## Statistics: Regression Setting

In statistics, non-robust estimators can be extremely sensitive to deviations from the assumptions. Indeed, inference is based not only on the sampled observations but also on the assumptions about their underlying properties. While these assumptions are not supposed to be exactly true, a minor error in the model "should cause only a small error in the final conclusions." [14] Tukey, among others, pointed out that many common statistical methods are excessively sensitive to "seemingly minor deviations from the assumptions." [14] Apart from statistical inference, the parameter vector itself in the regression setting defined in 1.1 can be arbitrarily skewed with just a single outlier in the data.

Consequently, a formal framework for modelling contaminated data was proposed by [13].

**Definition 1.5.** (Huber $\epsilon$-contamination model).
There is an underlying distribution $P_u$ and an outlier distribution $P_e$. All sampled observations are then sampled from the mixture distribution

$$\mathcal{D} = \big(1 - \epsilon\big) P_u + \epsilon P_e$$

where the goal is to learn the properties of the "good" distribution $P_u$. Notably, in this framework the set of outliers and the number of outliers are random.

Various robust estimators have been designed for learning from this contamination model, such as M-estimators or notions of depth. What they all have in common, however, is that they either work only in low-dimensional settings (Theil-Sen and Siegel estimator [21]), their error grows with the dimensionality of the estimation problem, or that they are computationally intractable. Huber in 1997 noted that "it is one thing to design a theoretical algorithm whose purpose is to prove [that it can tolerate a large fraction of outliers] and quite another thing to design a practical version that can be used not merely on small, but also on medium sized regression problems, with a 2000 by 50 matrix or so. This last requirement would seem to exclude all of the recently proposed [robust estimators]." [13]

There thus seem to be several and potentially conflicting objectives that a robust estimator should satisfy to be deemed "good": It should be computationally efficient, robust in "the sense that small deviations from the model assumptions should impair the performance only slightly", but also be robust to large deviations in what Huber terms "breakdown" [14] - the largest fraction of outliers a model can handle without allowing for arbitrarily skewed parameters. Robust estimators can thus be seen as compromising some, but as little as possible, efficiency for increased robustness. Ideally, a "good" robust estimator would also return an estimate close to that of a classical estimator in the absence of deviations from the assumptions.

It is noteworthy that the "classical" study of robust estimation placed importance more so on robustness against a few gross deviations than a systemic corruption of a potentially large fraction of the data, and also dealt mainly with estimation in low or medium dimensionality.

## Machine Learning: Mean Estimation & Stochastic Optimization

In machine learning research, in the meantime, the focus has been on making existing, applied approaches robust, usually done by designing practical and robust mean estimators (setting 1.2) which can then be applied to stochastic optimization (setting 1.3) in a batch learning setting. In a batch learning setting, the goal is still to minimize the risk, i.e. the expected error. However, the parameter values are iteratively updated using slices of the data, rather than the entire data set at once. This application will be shown in section 3.

Given that machine learning and statistical techniques are being used in increasingly sensitive domains but are vulnerable to biased or malicious data, the problem of robust estimation has recently drawn renewed attention in the computer science and machine learning community under the term "adversarial learning", for instance in [2, 19, 16], with the goal of developing estimators that work under minimal assumptions, in high dimensions and with low computational complexity.

To that end, the $\epsilon$-contamination model from definition 1.5 has been increasingly relaxed, yielding a much stronger adversarial contamination model [1]:

**Definition 1.6.** (Adversarial $\epsilon$-contamination model).
At first, samples are drawn from the underlying distribution $P_u$. An adversary can then select *any* $n\epsilon$ of the drawn points and replace them with *any* other points. The adversary is also given information on the model class, training algorithm, hyperparameters and employed defenses. [7, 24] The resulting set of samples is called *$\epsilon$-corrupted*.

Some of the first computationally tractable mean estimators robust to an $\epsilon$-fraction of adversarial outliers have been developed by [6] under the assumption of sub-Gaussian data. Since then, there has been a "flurry of research activity on algorithmic robust estimation in a variety of high-dimensional settings," [7] revisiting the classical robust statistics setting with a focus on "establishing [...] information theoretic dependencies between the fraction of corrupted data, dimension of the problem, and achievable accuracy of returned [...] estimate, and [...] developing computationally tractable algorithms that approach or achieve these information theoretic bounds." [19]

While much research has been done in settings where algorithms are designed for specific use-cases only, recently various general robust estimators have been

| **Algorithm** | **Error ($\delta$)** | **Runtime** | **Assumptions** |
|---|---|---|---|
| Dimension Halving[15] | $O(\epsilon\sqrt{\log d})$ | $\Omega(nd^2)$+ SVD | bounded 4th moment |
| Convex Programming[6] | $O(\epsilon\sqrt{\log(1/\epsilon)})$ | $O(n^4)$ | bounded 2nd moment |
| Filtering[6] | $O(\epsilon\sqrt{\log(1/\epsilon)})$ | $\Omega(nd^2)$ | bounded 2nd moment |

Table 1: Robust mean estimation algorithms and their properties[3], where $n$ denotes sample size, $d$ dimension and $\epsilon$ the fraction of outliers.

published that use robust mean estimation to obtain robust stochastic optimization algorithms. They work in the batch learning setting and rely only on the assumption of bounded moments. [5] use iterative scoring and subsequent removal of large gradients and [15] introduce a recursive, singular value decomposition (SVD) based algorithm, realizing that, in a high-dimensional mean estimation problem, knowing the direction of the mean shift essentially reduces the estimation problem to a 1-dimensional one. Further, in a more general setting, [23] define a property termed resilience - the mean of every large subset of some set being close to the overall mean - that enables robust mean estimation in high dimensions, and [18] consider robust probabilistic inference from a Bayesian viewpoint.

Table 1 displays the error bounds, sample complexities and run times of some of these results.

While classical robust statistics did not necessarily aim to make the breakdown point - the largest fraction of outliers an algorithm can handle - as large as possible, many of the recently developed algorithms can handle an outlier fraction up to 0.5. Additionally, frameworks have been proposed that allow learning in the presence of a majority of outliers. [23] Further, new developments specifically aim to achieve low error even in high dimensions to make the algorithms suitable for modern, applied machine learning. [2]

## Adversarial Online Learning

In the case of online learning, this is different. Currently there are no online learning algorithms robust to adversarial outliers.

Although most research regarding robustness has focused on offline learning, it is equally likely for an algorithm to face adversarial outliers in an online setting, and recent research showcases how to attack online learning algorithms. [24] focus on attacks against online learning algorithms in the case where the adversary knows the entire data stream beforehand. Different attack scenarios are given in [25] who consider the plausible case of an adversary having access, at any iteration, to the data stream up to and including that iteration. Notably, and similarly to stochastic optimization, without *any* defense such as ignoring large gradients that exceed a user-specified value, online learning algorithms can obtain arbitrarily large regret with just a single outlier.

Related to online learning, [17] present an algorithm robust to adversarial outliers in the stochastic bandit setting. Robustness is achieved by using several multi-armed bandits. One is trained on the entire data stream, while another is trained on a small, random sample of the data stream. If the fraction of outliers is small, that multi-armed bandit will likely see less outliers than the one that is trained on the entire data stream. It is thus robust to outliers because it rarely encounters them. Letting the arm eliminations of the robust multi-armed bandit apply also to the one trained on the entire data stream then yields a robust algorithm.

While no equivalent algorithm has been developed for the full feedback case yet, we show that the algorithms introduced in chapter 5 are somewhat similar and the approach of "robustness by simply not seeing outliers" is valid.

Further, none of the two contamination frameworks defined above can be straightforwardly applied to the online learning framework where data is processed sequentially. As such, we will also try to adapt the contamination framework to the online learning setting as well as propose a measure of regret that works in said framework.

## Outline

The remainder of the thesis consists of two parts. First, we review robust methods relying on i.i.d. assumptions, among them robust mean estimation and robust stochastic optimization. Then, we consider the case of adversarial online learning. We show why the ideas from the earlier chapters do not transfer and develop new approaches suited for data without the i.i.d. assumption.

Specifically, in the following chapter 2 we introduce the setting of robust mean estimation and consider some basic algorithms. We also consider the problem of translating these approaches into higher-dimensional settings. Thereafter, we consider to what extent these algorithms can carry over to the robust stochastic optimization setting in chapter 3. We show that, in fact, robust stochastic optimization can be seen as a robust mean estimation problem and, thus, all previously introduced algorithms apply.

Lastly, we deal with the problem of online learning in the presence of adversarial outliers. In chapter 4 we propose contamination models adjusted to the sequential nature of online learning and further propose robust regret, a performance measure for online learning algorithms in a contamination setting. We consider ways of making online learning robust to adversarial outliers and show that by using algorithms that minimize adaptive regret - i.e. algorithms that perform well even if the underlying data structure changes - we can obtain low robust regret with relatively low computational overhead. We also analyze how many outliers these algorithms can handle.

In chapter 5 we introduce specific algorithms for minimizing adaptive as well

as strongly adaptive regret, go through how the proofs are set up and conclude that, while relying on boundedness of the "good" data points, the algorithms do not require *any* assumptions regarding the adversarial outliers.

Finally, we discuss practical implementations and compare the performance of the introduced algorithms as well as current standard online learning algorithms on a simulated data set for regression in chapter 6.

# 2 A First Pass: Robust Mean Estimation

In this chapter we review some basic robust mean estimation algorithms that highlight the challenges of estimation in the face of $\epsilon$-contamination.

## 2.1 Framework

We consider a model where there is some true distribution $\mathcal{D}$ over $\mathcal{X} \subseteq \mathbb{R}^d$ with mean $\mu$ and we sample $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ i.i.d. points from that distribution. Our goal is then to estimate the mean such that our estimate is close to the true mean. In the absence of outliers, this is often done by taking the sample mean.

This is not a good estimator, however, under the adversarial $\epsilon$-contamination model defined in 1.6: After inspecting the data, an adversary can replace an $\epsilon n$ fraction of it with some other, arbitrary data. Clearly, in this case a single outlier suffices to change the sample mean to any arbitrary value. Our goal is thus to estimate the true mean $\mu$ via some estimate $\hat{\mu}$ such that $\|\hat{\mu} - \mu\|$ is small in some norm.

## 2.2 Robust Mean Estimation

### 2.2.1 Robust Estimation in 1 Dimension

Given that $d = 1$, one of the most basic results in robust mean estimation might be that of the trimmed mean. Assuming bounded variance $\mathbb{E}[(x - \mu)^2] \leq \sigma^2$ and given an $\epsilon$-fraction of outliers, we can obtain a robust estimate of the true mean by simply removing the most extreme values.

**Algorithm 1:** Trimmed Mean ((1) in [22])

1 Remove smallest and largest $2\epsilon n$ points (so $4\epsilon n$ points are removed in total).
2 Return the mean of the remaining points.

**Lemma 2.1.** ((1.2) in [22]). If $\mathcal{D}$ has mean $\mu$ and variance $\sigma^2$, and if $\epsilon \leq \frac{1}{8}$, algorithm 1 outputs an estimate $\hat{\mu}$ such that $|\hat{\mu} - \mu| \leq 8\sigma\sqrt{\epsilon}$.

The proof follows from proposition 1.2. in [22].

### 2.2.2 Robust Estimation in Higher Dimensions

Bounded variance also allows for the more challenging application of robust mean estimation in higher dimensions. As noted in [22], high dimensionality complicates robust estimation: Whereas in one dimension algorithms can easily

detect outliers based on their individual distances from the mean, in high dimensions outliers are often "no further from the mean than the good points in $\ell_2$-distance." Suggestions have been to e.g. extend the median-based approaches via the geometric median (see e.g. [9]). However, [15] show that geometric median based approaches have a $\sqrt{d}$ dependence on the dimension, rendering it "impractical" for many high-dimensional applications.

As such, to avoid incurring high error bounds dependent on dimensionality, robust estimation must "analyze entire populations of outliers at once" rather than look at data points in isolation. [22] This is consequently the approach that most of the recent robust estimation algorithms, e.g. those listed in table 1, take: They assign some weights to each observation, compute some estimation, update the weights based on some scoring process and reiterate.

For example, for arbitrary $d$, we can estimate $\mathbb{E}[\mathbf{x}]$ within $O(\sigma\sqrt{\epsilon})$ via algorithm 2 following [23] and [5], once again assuming bounded variance. We compare the empirical variance of each data point to some upper bound and down-weight points too far from the empirical mean.

**Algorithm 2:** FilterL2 [23, 5]

1 **Input** weights $c_1, \ldots, c_n = 1$
2 Compute $\hat{\mu}_c = \big(\sum_{i=1}^n c_i x_i\big) / \big(\sum_{i=1}^n c_i\big)$
3 Compute $\hat{\Sigma}_c = \sum_{i=1}^n c_i \big(x_i - \hat{\mu}_c\big)\big(x_i - \hat{\mu}_c\big)^T / \sum_{i=1}^n c_i$
4 Let $v$ be the largest eigenvector of $\hat{\Sigma}_c$ and let $\hat{\sigma}_c^2 = v^T \hat{\Sigma}_c v$
5 **if** $\hat{\sigma}_c \leq 16\sigma^2$ **then**
6 | **return** $\hat{\mu}_c$
7 **else**
8 | Let $\tau_i = \langle x_i - \hat{\mu}_c, v\rangle^2$
9 | Update $c_i \leftarrow c_i \cdot \big(1 - \tau_i / \tau_{\max}\big)$ where $\tau_{\max} = \max_i \tau_i$
10 | Return to line 2

The proof then also follows [23] and [5]. Intuitively, whenever the weighted variance $\hat{\sigma}_c^2$ is larger than the variance of the "good" data points by some constant, the outliers must be far away from the mean and can be down-weighted subsequently.

While it is thus possible to robustly estimate the mean in a high-dimensional setting, the implication is also that these algorithms cannot be made streamable and consequently cannot be applied in an online learning setting where we have to process each data point individually.

In the following chapter we describe how robust mean estimation can be applied to stochastic optimization to make it robust to scenarios where the sampled data points are contaminated by an adversary.

# 3 Robust Stochastic Optimization

We consider stochastic optimization algorithms under adversarial $\epsilon$-contamination and show how these algorithms can be made robust by applying robust mean estimation algorithms.

## 3.1 Framework

We consider the learning model where there is some true distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ and we sample $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ points from the data generating distribution. Our goal is then to find a parameter vector $\mathbf{h} \in \mathcal{H}$, $\mathcal{H} \subseteq \mathbb{R}^d$, where $\mathcal{H}$ denotes the parameter space, that minimizes the risk function $\mathbb{E}\big[\mathcal{L}(\mathbf{h}, \mathbf{x}, y)\big]$, where $\mathcal{L}$ is some loss function convex in $\mathbf{h}$. For example, often $\mathcal{L}(\mathbf{h}, \mathbf{x}, y) = (y - \mathbf{x}^T\mathbf{h})$.

Most stochastic optimization algorithms assume that all of the sampled data is generated by the underlying distribution. However, here we assume the adversarial $\epsilon$-contamination model defined in 1.6.

## 3.2 Robust Stochastic Optimization as Robust Mean Estimation

When splitting the data into batches, stochastic optimization can reduce to mean estimation, and thus the streamable algorithms could be applied to make stochastic optimization robust to outliers.

Steinhardt [22] and Diakonikolas et al. [7] also use their filter algorithm 2 to apply robust mean estimation to stochastic optimization, i.e. to find a $\gamma$-approximate stationary point of $\nabla_{\mathbf{h}}\mathbb{E}\big[\mathcal{L}(\mathbf{h}, \mathbf{x}, y)\big]$. In the following we let $\nabla = \nabla_{\mathbf{h}}$.

**Definition 3.1.** ($\gamma$-approximate stationary point). Given some parameter vector $\hat{\mathbf{h}}$ and the weights $c_i$ from algorithm 2, a $\gamma$-approximate stationary point satisfies

$$\left\| \sum_{i=1}^{n} c_i \nabla \mathcal{L}(\hat{\mathbf{h}}, \mathbf{x}_i, y_i) \Big/ \sum_{i=1}^{n} c_i \right\| \leq \gamma \tag{1}$$

Using algorithm 2, [22] devise algorithm 3 for robust stochastic optimization.

Robust mean estimation is thus performed on the gradients of the loss function w.r.t. the parameters, which leads to finding an approximate stationary point of the empirical risk function. Iteratively, a $\gamma$-approximate stationary point is found using some non-robust stochastic optimization algorithm. Then, "rather

**Algorithm 3:** RobustSO [(6) in [22]]

1 Initialize weights $c_1, \dots, c_n = 1$

2 Find any $\gamma$-approximate stationary point, i.e. any point $\hat{\mathbf{h}}$ s.t.

$$\| \sum_{i=1}^{n} c_i \nabla \mathcal{L}(\hat{\mathbf{h}}, \mathbf{x}_i, y_i) / \sum_{i=1}^{n} c_i \|_2 \leq \gamma$$

3 Run algorithm 2 with weights $c_i$ and $x_i = \nabla \mathcal{L}(\hat{\mathbf{h}}, \mathbf{x}_i, y_i)$

4 If algorithm 2 updates $c_i$ (see algorithm 2, line 9), let $c_i$ be the updated values and go back to line 2

5 Otherwise, output $\hat{\mathbf{h}}$

than re-initializing the weights $c_i$ to 1 every time [...], [they] are updated persistently" [22] while the algorithm is re-run. This is the same principle used in [7], who devise a meta-algorithm that can take in any base learner (e.g. linear regression) and make it robust to outliers by re-running it and in each iteration updating the weights of the data points.

As an alternative to iteratively weighting all data points, as done in the approach shown above, one could also consider the case of stochastic gradient descent, where minimizing the empirical loss function is an iterative process of the form

$$\mathbf{h}_{i+1} = \mathbf{h}_i - \eta \nabla \mathcal{L}\big(\mathbf{h}_i, \mathbf{x}_i, y_i\big)$$

for $i = 1, \dots, n$ and some step size $\eta > 0$. Noting that we can also perform stochastic gradient descent by grouping the $n$ samples into $b$ batches of size $m = n/b$, an update of our parameter vector changes to

$$\mathbf{h}_{i+1} = \mathbf{h}_i - \eta \nabla \bar{\mathcal{L}}\big(\mathbf{h}, \mathbf{x}, y\big) = \mathbf{h}_i - \eta \frac{1}{m} \sum_{j=(i-1)m+1}^{im} \nabla \mathcal{L}\big(\mathbf{h}_j, \mathbf{x}_j, y_j\big)$$

for $i = 1, \dots, b$. As can be seen, robust stochastic estimation thus reduces to robust mean estimation: If we can robustly estimate the mean gradient vectors $\nabla \mathcal{L}(\mathbf{h}, \mathbf{x}, y)$ of the batches, we can perform robust stochastic optimization. The goal is thus to robustly estimate, at each iteration of parameter updates, the mean gradient vector $\mathbb{E}[\nabla \mathcal{L}\big(\mathbf{h}_i, \mathbf{x}_i, y_i\big)]$ such that the distance between it and the true mean $\mu$, $\|\mathbb{E}[\nabla \mathcal{L}\big(\mathbf{h}_i, \mathbf{x}_i, y_i\big)] - \mu\|$, is small in some norm.

This can be done using any of the algorithms displayed in this and previous chapters. However, since the algorithm is not constantly re-run with re-weighted

data points as is done in e.g. algorithms 2 and 3 but rather used with approximate batch estimates, the error would be proportional to the number of batches the data is split into. [7]

# 4 Robust Online Learning

In this chapter we study the problem of online learning in adversarial $\epsilon$-contamination scenarios. We propose a framework for online contamination scenarios and discuss why the current approaches of robust mean estimation and robust stochastic optimization do not apply. Thereafter, we introduce algorithms that can learn (online) under the introduced contamination settings and also consider their computational complexity.

## 4.1 Framework

We consider the online learning framework where some data $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\} \in \mathcal{X} \times \mathcal{Y}$ is received over time. We make no probabilistic assumptions regarding the origins of the data. At every time point $t$, the online player receives an entry $\mathbf{x}_t$ from the data stream and makes a decision $h \in \mathcal{H}$ where $\mathcal{H}$ is a bounded convex set with diameter at most $D$ denoting the decision space. Then, the online player receives the true answer $y_t \in \mathcal{Y}$ and suffers loss via the convex loss function $f_t : \mathcal{H} \mapsto \mathbb{R}$. In particular, we will instantiate this to the case where $f_t(h) = \mathcal{L}(h(\mathbf{x}_t), y_t)$, where $h_t(\mathbf{x}_t) = \hat{y}_t$, the choice of the online player at round $t$. Commonly, this is e.g. $h_t(\mathbf{x}_t) = \mathbf{x}_t^T h,\ h \in \mathbb{R}^d$. [20]

Instead of minimizing the risk function as in the stochastic optimization framework, the performance of online learning algorithms is measured in terms of regret: the difference between the total loss incurred and the best fixed decision a player could have taken in hindsight. [20]

**Definition 4.1.** (Regret)
The regret of an online optimization algorithm $\mathcal{A}$ after $T$ iterations w.r.t. a hypothesis class $\mathcal{H}$, where each hypothesis $h^* \in \mathcal{H}$ maps to the target domain, $h^* : \mathcal{X} \to \mathcal{Y}$, is defined as the cumulative difference between the incurred loss and the loss of the best fixed decision in hindsight:

$$\text{Regret}_T(\mathcal{H}) = \max_{h^* \in \mathcal{H}} \left\{ \sum_{t=1}^{T} \mathcal{L}(\hat{y}_t, y_t) - \sum_{t=1}^{T} \mathcal{L}\big(h^*(x_t), y_t\big) \right\} \tag{2}$$

The goal of an online learning algorithm is then to obtain the lowest possible regret w.r.t. a hypothesis class $\mathcal{H}$.

Unlike in the traditional online optimization setting, we also propose a contamination model. The existing contamination models defined in 1.6 suited for offline optimization do not take into account the sequential nature of online learning. As such, we propose a new framework adjusted to online learning that we term *online adversarial $\epsilon$-contamination model.*

**Definition 4.2.** (Online adversarial $\epsilon$-contamination model)
At any time point $t$, an adversary can corrupt the data point $(\mathbf{x}_t, y_t)$. Then, we propose two possible scenarios:

1. An adversarial $\epsilon$-contamination model: The adversary gets to replace an $\epsilon$ fraction of the entire data stream. At any time point $t$, $\min\{t, \epsilon T\}$ of the data points could be corrupted.
2. A time-constant adversarial $\epsilon$-contamination model: An adversary gets to replace an $\epsilon$-fraction of the data *over time*. At any time point $t$, $t > 1$, $\epsilon t$ of the data could be corrupted.

Given adversarial outliers, the performance can no longer be measured by considering the loss for all points in the data stream. Rather, we would ideally want to devise methods that achieve low regret for the "good" data points and that are only marginally influenced by the adversarial outliers. Therefore, we define a new performance measure termed "Robust Regret":

**Definition 4.3.** (Robust Regret)
Let $\mathcal{I}$ denote the indices of the $(1-\epsilon)T$ "good" data points. Then, similar to definition 4.1, the robust regret of some online optimization algorithm $\mathcal{A}$ after $T$ iterations w.r.t. a hypothesis class $\mathcal{H}$ is defined as the cumulative difference between the incurred loss and the loss of the best fixed decision in hindsight, considering only the "good" data points:

$$\text{Robust-Regret}_T(\mathcal{H}, \mathcal{I}) = \max_{h^* \in \mathcal{H}} \left\{ \sum_{t \in \mathcal{I}} \mathcal{L}(\hat{y}_t, y_t) - \sum_{t \in \mathcal{I}} \mathcal{L}\big(h^*(x_t), y_t\big) \right\} \tag{3}$$

Robust regret applies to both of the $\epsilon$-contamination scenarios mentioned above.

## 4.2 Robust Online Learning via Robust Batch Algorithms

An intuitive approach to making online learning robust to the added outliers would be to borrow from the robust mean estimation and robust stochastic optimization approaches described in earlier chapters. While it might be feasible to split the data stream into batches and use robust batch estimates for online learning, this is complicated by other factors.

Algorithms from robust stochastic optimization cannot simply be applied to online learning scenarios given that stochastic optimization makes the assumption that the sampled data is drawn i.i.d. from some distribution $\mathcal{D}$. Online learning on the other hand makes no assumptions on the origins of the data. It could be deterministic, stochastic or even adversarially adaptive to the learning algorithm, and it is this worst case that online learning algorithms tend to be designed for. Thus, robust stochastic optimization does not quite fit into the adversarial online learning framework.

Further, the first scenario - $\epsilon T$ possible outliers at any time point $t$ - complicates online learning significantly. Intuitively, no stochastic optimization algorithm can obtain a breakdown point better than $\frac{1}{2}$. If the majority of data consists of adversarial outliers, it might be the good data points that are thought to be outliers, and the outliers that are treated as being from some good distribution to estimate. In fact, [8] show that "no translation-equivariant estimator can achieve a breakdown point better than $\frac{1}{2}$." [23] As such, our batch estimates would be based on the outliers and the online learning algorithm would not be able to recover. Instead, applying algorithms from robust stochastic optimization would imply having to first sample $2\epsilon T+1$ data points. After all, we cannot rule out the case of all outliers being at the start of the data stream, meaning that this is the amount of data needed to have a majority of 'good' data.

It is for these reasons that new approaches are necessary. An alternative approach well-suited to online learning is to not use batches and try to make algorithms "recover" from the influence of outliers, i.e. return to a normal state, as quickly as possible. This is the approach we describe in the following sections.

### 4.3 Robust Online Learning via Leaving Out Data Points

An intuitive way of making an algorithm more robust is to train the algorithm on all points but the corrupted ones. However, in practice we do not know which data points are corrupted and which ones are 'good'. If we do know, however, how many data points are corrupted in total, then we can consider leaving out all possible combinations of data points. Running one algorithm per combination, and combining the algorithms' predictions based on their respective performances can then yield a robust online learning algorithm, as described in algorithm 4.

This algorithm is an altered version of the popular multiplicative weights algorithm. The multiplicative weights algorithm uses experts - i.e. standard online learning algorithms - that each leave out different combinations of data points. At any time point $t$, the algorithm's prediction is a convex combination of the (active) individual experts' predictions, based on the experts' weights which in turn are based on their respective instantaneous regret, as defined in 4.4 .

**Definition 4.4.** (Instantaneous Regret)
Denote by $\mathcal{L}_t$ the full loss of a meta-learning algorithm at time point $t$. Further, let $\mathcal{L}_t^{(i)}$ denote the loss of the $i$th base learner at time point $t$. The instantaneous regret of the algorithm w.r.t. this base learner at time point $t$ is then

$$\mathcal{R}_t^{(i)} = \mathcal{L}_t - \mathcal{L}_t^{(i)}$$

i.e. the difference between the loss of the full algorithm and the loss of the base learner.

**Algorithm 4:** n-Robust Multiplicative Weights

**Input:** step size $\eta$, # of outliers $\lceil \epsilon T \rceil$.
Set $S = \left\{ A \subseteq \{1, \dots, T\} \mid |A| = \lceil \epsilon T \rceil \right\}$, the set of all combinations of data points to leave out.
Let $E^{(1)}, \dots, E^{(|S|)}$ be online convex optimization algorithms.
Initialize weights $\{w_i\}_{i=1}^{|S|} = 1/\binom{T}{\lceil \epsilon T \rceil}$.
**for** $t = 1$ **to** $T$ **do**
    Set the active set of experts, Active $= \{i \mid t \notin S_i\}$
    Set $\forall_{j \in \text{Active}},\ \hat{y}_t^{(j)} \leftarrow E^{(j)}(x_t)$ (the prediction of the $j$th algorithm)
    Predict a convex combination of the experts, i.e. $\hat{y}_t = \sum_{\text{Active}} \hat{y}_t^{(i)} \frac{w_i}{\sum_{\text{Active}} w_j}$
    Receive loss $\mathcal{L}_t^{(i)}$ per active expert
    For $i \in$ Active perform update:

$$w_i = w_i \cdot (1 + \eta \mathcal{R}_t^{(i)})$$

    where $\mathcal{R}_t^{(i)}$ is the instantaneous regret of the $i$th expert at time point $t$

**Theorem 4.1.** (Regret Bound for Algorithm 4)
For a data stream $\{1, \dots, T\}$, given a known number $\lceil \epsilon T \rceil$, of corrupted data points, step size $\eta \in [0, \frac{1}{2}]$ and bounded loss $\mathcal{L}_t^{(i)} \in [0, 1]$, algorithm 4 achieves robust regret at most

$$\max_{h^* \in \mathcal{H}} \left\{ \sum_{t \in \mathcal{I}} \mathcal{L}(\hat{y}_t, y_t) - \sum_{t \in \mathcal{I}} \mathcal{L}\big(h^*(x_t), y_t\big) \right\} \leq \eta^{-1} \ln T + \eta T + \eta^{-1} + \eta^{-1} \lceil \epsilon T \rceil \ln T + R(T)$$

where $R(T)$ is the regret of the best base learner (the algorithm that learns only on the good data points) at time point $T$.

The proof to this theorem is shown in appendix A. The resulting robust regret bound works in that e.g. for step size $\eta = \frac{1}{\sqrt{T}}$, it reduces to $O(\sqrt{T} \lceil \epsilon T \rceil \ln T) + R(T)$. We must then have that $\lceil \epsilon T \rceil \ll \sqrt{T}$ to obtain a useful (sublinear) regret bound. However, given that we need to consider *all* $\binom{T}{\lceil \epsilon T \rceil}$ combinations of data points to leave out, there is a large computational overhead of $\binom{T}{\lceil \epsilon T \rceil} \leq \left(\frac{eT}{\lceil \epsilon T \rceil}\right)^{\lceil \epsilon T \rceil}$.

This means that the algorithm can become practically unfeasible as soon as there is more than a single outlier, and as such we need to consider alternatives that are more computationally efficient, while still providing sublinear robust regret bounds. One such approach is to use adaptive online learning algorithms, as detailed in the following section.

### 4.4 Robust Online Learning via Adaptive Regret

The online learning framework is no stranger to algorithms devised for learning despite changes in the underlying distributions by letting the algorithm adapt to said changes. The idea is to devise algorithms that not only perform well on the entire data stream, but also on smaller intervals of the data stream while adding little computational overhead. For this, Hazan and Seshadhri in [12] introduce *adaptive regret* defined as:

**Definition 4.5.** (Adaptive Regret, [1.1] in [12])

$$\text{Adaptive-Regret}_T(\mathcal{H}) = \sup_{I=[r,s]\subseteq[T]} \max_{h^*\in\mathcal{H}} \left\{ \sum_{t\in I} \mathcal{L}(\hat{y}_t, y_t) - \sum_{t\in I} \mathcal{L}\big(h^*(x_t), y_t\big) \right\}$$

I.e. "the maximum regret [some algorithm $\mathcal{A}$] achieves over any contiguous time interval" [12] w.r.t. a hypothesis class $\mathcal{H}$. Unlike online learning under the classic definition of regret, adaptive regret allows for the learning of changes in the distributions since the loss of the algorithm over some interval is compared to the loss obtained via the best fixed decision in hindsight *for that interval*.

Then, we can think of the adversarial outliers as splitting our data into intervals of good data. Given that an adversarial outlier can change the parameter vector of an online learning algorithm (arbitrarily in the case of unbounded outliers) such that the performance on all the following data points is negatively impacted, we could require a robust online learning algorithm to perform well on the good data preceding the outlier and the good data following the outlier.

An adversary that can corrupt an $\epsilon$ fraction of the data stream can split the stream into $J \leq \epsilon T + 1$ intervals of good data. We denote by $\mathcal{K}$ the *set* of intervals of good data, i.e. $\mathcal{K} = \{I_i\}_{i=1}^{J}$. Adaptive regret can then be used to make online learning algorithms robust by bounding the regret on each of these intervals. Summing up the regret of each interval of good data, we obtain the robust regret as shown below:

$$\begin{aligned}
\text{Robust-Regret}_T(\mathcal{H}, \mathcal{I}) &= \max_{h^*\in\mathcal{H}} \left\{ \sum_{t\in\mathcal{I}} \mathcal{L}(\hat{y}_t, y_t) - \sum_{t\in\mathcal{I}} \mathcal{L}\big(h^*(x_t), y_t\big) \right\} \\
&= \max_{h^*\in\mathcal{H}} \sum_{i=1}^{J} \left\{ \sum_{t\in I_i} \mathcal{L}(\hat{y}_t, y_t) - \sum_{t\in I_i} \mathcal{L}\big(h^*(x_t), y_t\big) \right\} \\
&\leq \sum_{i=1}^{J} \max_{h^*\in\mathcal{H}} \left\{ \sum_{t\in I_i} \mathcal{L}(\hat{y}_t, y_t) - \sum_{t\in I_i} \mathcal{L}\big(h^*(x_t), y_t\big) \right\} \\
&\leq J \cdot \max_{I\in\mathcal{K}} \max_{h^*\in\mathcal{H}} \left\{ \sum_{t\in I} \mathcal{L}(\hat{y}_t, y_t) - \sum_{t\in I} \mathcal{L}\big(h^*(x_t), y_t\big) \right\} \\
&\leq J \cdot \text{Adaptive-Regret}_T(\mathcal{H})
\end{aligned}$$

where the two last terms denote $J$ times the adaptive regret *when only considering* $\mathcal{K}$*, the intervals of good data*, and the adaptive regret for the entire data stream, respectively.

Assuming there is some algorithm $\mathcal{A}$ that can achieve adaptive regret Adaptive-Regret$_T(\mathcal{H})$ w.r.t some hypothesis class $\mathcal{H}$ and when considering only $\mathcal{K}$, the intervals of good data, then we know that we can guarantee at most that regret w.r.t. *any* interval $I \in \mathcal{K}$. Regret bounds are only non-vacuous if they are sublinear in $T$. We therefore must have that $J \cdot$ Adaptive-Regret$_T(\mathcal{H}) = o(T)$. Since Adaptive-Regret$_T(\mathcal{H}) = O(\sqrt{T}\ln T)$, as we describe in chapter 5, and $J$ in the worst case $J = \epsilon T + 1$, we must have that the number of outliers $\epsilon T \ll \sqrt{T}$ to obtain a useful (sublinear) robust regret bound. Interestingly, this is similar to the result obtained for robustness via leaving out combinations of data points in algorithm 4 and theorem 4.1. However, the computational overhead of adaptive algorithms is much smaller as we describe in chapter 5.

**Robustness via Strongly Adaptive Regret**

All existing adaptive regret algorithms are meta algorithms in that they take an existing online learning algorithm (base learner) and turn it into an adaptive regret algorithm. For their analyses, all adaptive regret algorithms assume bounded losses. For an interval $I = [r, s]$ on the data stream, the adaptive algorithm presented in [12] has adaptive regret at most $O(\sqrt{s}\ln s) + R(s)$ where $R(s)$ is the regret of the base learner used. This is not yet ideal. A term depending on $s$, i.e. the end of our interval, implies that the adaptive regret depends on the size of the data stream, rather than just the interval size. It also implies that it can matter to an adversary whether he places outliers in the beginning or the end of the data stream. The most critical implication, however, is that these adaptive regret algorithms do not perform well on small intervals. Given e.g. an interval of size less than $\sqrt{T}$, a regret bound of $O(\sqrt{T})$ is vacuous. It is for this reason that [4] propose a stronger version of adaptive regret that depends on the length of the intervals.

In the following, we denote the size of an interval $I = [r, s]$ by $k = |I| = s - r + 1$.

**Definition 4.6.** (Strongly Adaptive Regret [4])
For an interval $I = [r, s]$ of size $k \subseteq [T]$ , the regret of an algorithm $\mathcal{A}$ during the interval $I$ is $R_{\mathcal{A}}(I)$. The *strongly adaptive regret* of $\mathcal{A}$ at time $T$ is the function

$$\text{SA-Regret}_T^{\mathcal{A}}(k) = \max_{I=[r,r+k-1]\subset[T]} R_{\mathcal{A}}(I)$$

We say that $\mathcal{A}$ is *strongly adaptive* if SA-Regret$_T^{\mathcal{A}}(k) = O(\text{poly}(\ln T) \cdot R_{\mathcal{P}}(k))$, where $R_{\mathcal{P}}(k)$ is the minimax regret for $k$ rounds in learning problem $\mathcal{P}$. [4]

Generally, strongly adaptive algorithms are desirable because they can handle a much larger number of outliers (given the fact that they can learn on intervals of any size) and make the adversary indifferent as to the positions in the data stream at which the outliers are placed, since the regret of an interval depends on the size of that interval, not on its position in the data stream.

Further, algorithms guaranteeing strongly adaptive regret can handle a larger number of outliers. We previously outlined how for adaptive regret algorithms, we require $\epsilon \ll \sqrt{T}$ to obtain useful robust regret bounds. Given the robust regret bound of the strongly adaptive learning algorithm 7, a good strategy for an adversary with the aim of maximizing robust regret is to place outliers such that the data stream is split into intervals of "good" data of even sizes, as we will prove below. Here, we use the regret bound $\text{SA-Regret}_{\mathcal{A}}^{T}(k) \leq C \ln T \sqrt{k}$ from algorithm 7 that we will introduce in the following chapter and where $C$ is some constant.

It follows then, also, that the strongly adaptive regret algorithm 7 can handle more outliers than the adaptive algorithm 6, both described in detail in chapter 5:

$$
\begin{aligned}
\text{Robust-Regret}_T(\mathcal{H}, \mathcal{I}) &\leq \sum_{i=1}^{J} \max_{h^* \in \mathcal{H}} \left\{ \sum_{t \in I_i} \mathcal{L}(\hat{y}_t, y_t) - \sum_{t \in I_i} \mathcal{L}(h^*(x_t), y_t) \right\} \\
&\leq \sum_{i=1}^{J} C \ln T \sqrt{|I_i|} \\
&= JC \ln T \left( \sum_{i=1}^{J} \sqrt{|I_i|} \right) / J \\
&\leq JC \ln T \sqrt{\frac{\sum_{i=1}^{J} |I_i|}{J}} && \text{Jensen's inequality} \\
&\leq C \ln T \sqrt{JT} && \text{using that } \sum_{i=1}^{J} |I_i| \leq T
\end{aligned}
$$

For $J = \epsilon T$ we see that now, with the strongly adaptive algorithm 7, we require $\epsilon T = o(T)$ rather than $\epsilon T = o(\sqrt{T})$ as shown for the adaptive algorithm 6, i.e. the fraction $\epsilon$ of outliers that the strongly adaptive algorithm can handle is significantly larger, and not far from the robust regret bound of algorithm 4 when $J$ is small.

In the following chapter we review algorithms for adaptive regret and strongly adaptive regret learning.

# 5 Robust Online Learning Algorithms

In this chapter, we introduce robust online learning algorithms. These are the (strongly) adaptive algorithms that were briefly outlined in the previous chapter. We divide the chapter into three subsections.

We first demonstrate how one can take an existing algorithm and turn it into an adaptive algorithm by separating the loss throughout the entire data stream $t = 1, \dots, T$ into the loss throughout an interval and the loss of data points not in that interval. This will provide a basis for understanding the following two algorithms.

We then introduce an adaptive regret algorithm by [12] in subsection 5.2 and a strongly adaptive one by [4] in subsection 5.3. Lastly, we compare the algorithms' properties and the respective implications for robust online learning.

## 5.1 A Basic Adaptive Algorithm

First, we show how one can take the standard multiplicative weights algorithm and turn it into an adaptive regret algorithm. This algorithm is detailed in algorithm 5.

**Algorithm 5:** Adaptive Multiplicative Weights

**Input:** step size $\eta$.
Let $E^{(1)}, \dots, E^{(T)}$ be online convex optimization algorithms.
Initialize weights $\{w_i\}^T = 1$.
**for** $t = 1$ **to** $T$ **do**
  Set $\forall j \leq t, \hat{y}_t^{(j)} \leftarrow E^{(j)}(x_t)$ (the prediction of the $j$th algorithm)
  Predict a convex combination of the experts, i.e. $\hat{y}_t = \sum_{i=1}^{t} \hat{y}_t^{(i)} \frac{w_i}{\sum_{j=1}^{t} w_j}$
  Receive loss $\mathcal{L}_t^{(i)}$ per expert $\{E^{(i)}\}_{i=1}^{t}$
  For $1 \leq i \leq t$ perform update:

$$w_i = w_i \cdot (1 + \eta \mathcal{R}_t^{(i)})$$

  where $\mathcal{R}_t^{(i)}$ is the instantaneous regret of the $i$th expert at time point $t$

The multiplicative weights algorithm uses experts - i.e. standard online learning algorithms - that start at different time points. There are $T$ experts starting at time points $1, \dots, T$, respectively. At any time point $t$, the algorithm's prediction is a convex combination of the individual experts' predictions, based on the experts' weights which in turn are based on their respective instantaneous regret, as defined in 4.4.

To show that our algorithm is adaptive, we have to compare the loss of the *full* algorithm to that of the *best* expert, i.e. the one that learns only on the interval of interest (i.e. the respective intervals containing only good data points). This is the regret w.r.t. the best expert, for which we will provide a bound in the following theorem.

**Theorem 5.1.** (Regret Bound for Algorithm 5)
For an interval $I = [r, s]$ of size $k = s - r + 1$, and with bounded loss $\mathcal{L}_t^{(r)} \in [0, 1]$ and step size $\eta \in [0, \frac{1}{2}]$, algorithm 5 achieves adaptive regret at most

$$\max_{h^* \in \mathcal{H}} \left\{ \sum_{t \in I} \mathcal{L}(\hat{y}_t, y_t) - \sum_{t \in I} \mathcal{L}(h^*(x_t), y_t) \right\} \leq \eta^{-1} \ln s + \eta k + R(I)$$

where $\mathcal{L}_t^{(r)}$ is the loss of the $r$th expert at time point $t$ and $R(I)$ is the regret of base learner (the $r$th algorithm, $E^{(r)}$) over the interval $I$.

*Proof.* For the proof we follow the common multiplicative weights proof using a potential function, with slight adaptations. We denote by $w_i^{(t)}$ the weight of the $i$th expert at time point $t$ and use the potential function $W(t) = \sum_{i=1}^{t} w_i^t$. Then we relate the sum of the weights at time point $s$, i.e. the end of the interval, to the performance of the best expert (the one who started at time point $r$) throughout the same interval.

As before we denote the loss of the full algorithm at time point $t$ with $\mathcal{L}_t$. Then, at time point $t$, we can think of all the experts $E^{(t)}, \ldots, E^{(s)}$, i.e. those that will be introduced at a later time point, as experts with constant weights incurring no loss.

For any $t$ we have that

$$\begin{aligned}
W(t+1) = \sum_{i=1}^{t+1} w_t^{(i)} &= 1 + \sum_{i=1}^{t} w_t^{(i)} \big(1 + \eta(\mathcal{L}_t - \mathcal{L}_t^{(i)})\big) \\
&\leq 1 + \sum_{i=1}^{t} w_t^{(i)} \left(1 + \eta\left(\sum_{j=1}^{t} \frac{w_t^{(j)} \mathcal{L}_t^{(j)}}{\sum_{j=1}^{t} w_t^{(j)}} - \mathcal{L}_t^{(i)}\right)\right) \quad \text{(by Jensen's inequality)} \\
&= 1 + W(t) + \eta\big(\sum_{j=1}^{t} w_t^{(j)} \mathcal{L}_t^{(j)} - \sum_{i=1}^{t} w_t^{(i)} \mathcal{L}_t^{(i)}\big) \\
&= 1 + W(t)
\end{aligned}$$

In particular, by induction, we have that $W(s+1) \leq s + 1$.

Further, considering the loss of the best expert (the one who started at the beginning of the interval, i.e. at r), $\mathcal{L}_t^{(r)}$, and the resulting instantaneous regret,

we know that the sum of the experts' weights must be at least

$$W(s+1) \geq 1 + \prod_{t=r}^{s}(1+\eta\mathcal{R}_t^{(r)})$$

$$\begin{aligned}
\ln s \geq \ln W(s+1) - 1 &\geq \sum_{t=r}^{s} \ln(1+\eta\mathcal{R}_t^{(r)}) \\
&\geq \sum_{t=r}^{s} \eta\mathcal{R}_t^{(r)} - \sum_{t=r}^{s}(\eta\mathcal{R}_t^{(r)})^2 \quad \text{since } \ln(1+x) \geq x - x^2 \text{ for } x \in [-1/2, 1/2] \\
&\geq \eta\big(\sum_{t=r}^{s} \mathcal{R}_t^{(r)} - \eta k\big)
\end{aligned}$$

As such,

$$\begin{aligned}
\eta\big(\sum_{t=r}^{s} \mathcal{R}_t^{(r)} - \eta k\big) &\leq \ln s \\
\sum_{t=r}^{s} \mathcal{R}_t^{(r)} &\leq \eta^{-1}\ln s + \eta k
\end{aligned}$$

Combining this with the regret of the base learner completes the proof. □

We note that we could achieve adaptive regret of $O(\sqrt{T}\ln T)$ by setting the step size $\eta = 1/\sqrt{T}$. This follows from simply plugging in the step size into the bound, yielding

$$\begin{aligned}
\sum_{t=r}^{s} \mathcal{L}_t - \mathcal{L}_t^{(r)} &\leq \sqrt{T}\ln(s+1) + \frac{k}{\sqrt{T}} \\
&\leq O(\sqrt{T}\ln T)
\end{aligned}$$

**Remark.** Ideally, we would of course want to let $\eta = \frac{1}{\sqrt{k}}$, i.e. let the step size depend on the interval size $k$. This would yield an even better, *strongly adaptive* bound. For obvious reasons, however, we do not know the size of the intervals we are looking for since that would require us to know the number of outliers as well as their positions in the data stream.

In the following we introduce two algorithms from the literature on adaptive and strongly adaptive regret.

## 5.2 Hazan and Seshadhri's Adaptive Algorithm

For adaptive regret learning, Hazan and Seshadhri [11] propose an algorithm termed "Follow the Leading History" (FLH) that achieves sublinear adaptive regret via a slight adaptation of the multiplicative weights algorithm. By reducing the "continuous optimization problem to the realm of discrete experts,

which are themselves online optimization algorithms," one can choose experts such that at least one is guaranteed to "have good performance at any game iteration." [12] This is surprisingly similar to the robust multi-armed bandit [17] mentioned in section 1: each expert represents an arm (although the feedback is obviously not bandit feedback), and robustness is achieved by starting arms at different time points.

At each time point $t = 1, \ldots, T$, a new expert (i.e. online learning algorithm) starts learning from that time point through to $T$. That way, for any interval $I = [r, s]$, there is some optimal algorithm that started learning at the beginning of said interval and is not influenced by past data points. Each expert's performance is tracked via a probability vector and the full algorithm's predictions are convex combinations of the experts' predictions weighted by their performance, as detailed in algorithm 6.

**Algorithm 6:** Follow the Leading History for Convex Loss Functions [(1) in [12]]

Let $E^{(1)}, \ldots, E^{(T)}$ be online convex optimization algorithms.
Initialize probability of first expert, $p_1 = 1$.
**for** $t = 1$ **to** $T$ **do**
  Denote by $p_t$ the probabilities for the first $t$ experts, $p_t = \{p_i\}_{i=1}^{t}$.
  Set $\forall j \leq t, \hat{y}_t^{(j)} \leftarrow E^{(j)}(x_t)$ (the prediction of the $j$th algorithm)
  Play $\hat{y}_t = \sum_{j=1}^{t} p_t^{(j)} \hat{y}_t^{(j)}$
  Receive loss function $\mathcal{L}_t$
  For $1 \leq i \leq t$ perform update:

$$\hat{p}_{t+1}^{(i)} = \frac{p_t^{(i)} e^{-\eta_t \mathcal{L}_t(\hat{y}_t^{(i)})}}{\sum_{j=1}^{t} p_t^{(j)} e^{-\eta_t \mathcal{L}_t(\hat{y}_t^{(j)})}}$$

  Set $p_{t+1}^{(t+1)}$ to $1/(t+1)$ and for $i \neq t+1 : p_{t+1}^{(i)} = (1 - (t+1)^{-1}) \hat{p}_{t+1}^{(i)}$

As shown in [11], for bounded general convex loss functions, $\mathcal{L}_t \in [0, M]$, algorithm 6 achieves adaptive regret of order $R(T) + \sqrt{T} \ln T$, where $R(T)$ is the regret achieved by the algorithms that make up the experts.

With a slightly more careful analysis we can improve Hazan and Seshadhri's adaptive regret bound for general convex loss functions. In [11] adaptive regret for some interval $I = [r, s]$ is shown to be at most $O(\sqrt{s} \ln s) + R(s)$, meaning an interval's adaptive regret depends on its position in the data stream rather than the size of the interval. With constants, the adaptive regret bound in [11] is

$$\sum_{t=r}^{s} \mathcal{L}_t - \mathcal{L}_t^{(r)} \leq 2\sqrt{r} \ln r + O(\sqrt{s} \ln s) + (M^2 + 2)(\sqrt{s} - \sqrt{r})$$

We can improve this slightly as shown in the following theorem.

**Theorem 5.2.** (Improved Adaptive Regret Bound for Algorithm 6)
With the same bounded loss, $\mathcal{L}_t \in [0, M]$ and for intervals $I = [r, s]$, algorithm 6 achieves adaptive regret at most

$$\max_{h^* \in \mathcal{H}} \left\{ \sum_{t \in I} \mathcal{L}(\hat{y}_t, y_t) - \sum_{t \in I} \mathcal{L}(h^*(x_t), y_t) \right\} \leq (\sqrt{2}+1)M + \sqrt{r+1} \ln r + O(\sqrt{k} \ln k) + (2M^2+4)\sqrt{k}$$

with step size $\eta_t = 1/\sqrt{t}$.

The proof to this theorem is shown in appendix B. The bound provided is a slight improvement of the original result in [11], replacing $O(\sqrt{s} \ln s)$ with $O(\sqrt{k} \ln k)$, i.e. the interval size. This implies that an adversary cannot increase robust regret by arbitrarily extending the interval of good data (or inserting the outliers only at the beginning of the data stream).

Even with this improvement, however, Hazan and Seshadhri's adaptive regret algorithm is not yet ideal since, in the worst case, $r$ is much larger than the interval size $k$. Then, like before, an adversary could place outliers towards the end of the data stream to maximize the robust regret bounds of the individual intervals created. The strongly adaptive algorithm introduced in the following subsection improves on this by making the bound depend only on the interval size $k$.

#### Computational Complexity

The Follow the Leading History (FLH) algorithm 6 takes an online base learner such as online gradient descent and initiates $T$ instances of that learner that are run on data stream intervals of length $1, \ldots, T$, respectively. Given observations $\{\mathbf{x}_i\}^T \in \mathbb{R}^d$, each algorithm at any time point can output predictions, compute gradients and update its weight vectors in $O(d)$ time. The time complexity of FLH is thus $O(T^2 d)$. More computationally efficient implementations are discussed in [11], albeit at the cost of slightly worse adaptive regret bounds.

### 5.3 Daniely et al.'s Strongly Adaptive Algorithm

For strongly adaptive regret learning and bounded losses in $[0, 1]$, [4] present an algorithm achieving SA-Regret of $O(\ln(s)\sqrt{k})$. Similar to algorithm 6, this algorithm is also a variant of the multiplicative weights algorithm. However, unlike the previous algorithms where one base learner was initialized at each time point $1, \ldots, T$, this algorithm utilizes a more intricate way of covering different intervals of data, as described below.

For each interval $I_i$ out of a set of chosen intervals, a base learner $E^{(i)}$ (an online learning algorithm) is run. Similar to the preceding algorithms, the base learners are then weighted based on their performance. At any time point $t$, the algorithm picks one of the base learners' predictions at random with probabilities

proportional to the experts' weights. Rather than via loss, the performance of the experts in the strongly adaptive online learning algorithm is measured via instantaneous regret, as previously defined in 4.4.

Further, each base learner has an individual step size $\eta_i$ based on the interval that respective base learner is covering

$$\eta_i = \min\{1/2, 1/\sqrt{|I_i|}\}$$

Also, experts are sampled in a specific way to keep the set of active experts (i.e. those whose learning interval includes time point $t$) small. Following [4]: The set of intervals that will each be covered by a base learner is

$$\mathcal{J} = \bigcup_{k \in \mathbb{N} \cup \{0\}} \mathcal{J}_k$$

where for all $k \in \mathbb{N} \cup \{0\}$,

$$\mathcal{J}_k = \{[i \cdot 2^k, (i+1) \cdot 2^k - 1] \mid i \in \mathbb{N}\}$$

Then we denote $\text{ACTIVE}(t) = \{i \mid t \in I_i\} \subseteq \mathcal{J}$, the set of indices of experts whose intervals include time point $t$.

The algorithm is shown in algorithm 7, where experts are denoted by $E^{(i)}$, and their respective weights at time point $t$ by $w_t^{(i)}$.

**Algorithm 7:** Strongly Adaptive Online Learner [(1) in [4]]

For all experts $i$ in ACTIVE(1), initialize $w_1^{(i)} = \begin{cases} 1/2 & \text{if } I_i = [1,1] \\ 0 & \text{otherwise} \end{cases}$

**for** $t = 1$ **to** $T$ **do**

  Let $W_t = \sum_{i \in \text{ACTIVE}(t)} w_t^{(i)}$

  Choose $E^{(i)},\ i \in \text{ACTIVE}(t)$ with probability $p_t^{(i)} = \frac{w_t^{(i)}}{W_t}$

  Predict $\hat{y}_t^{(i)}$

  Update weights $w_{t+1}^{(i)} = \begin{cases} 0 & t \notin I \\ \eta_i & t = r \\ w_t^{(i)}(1 + \eta_i \cdot \mathcal{R}_t^{(i)}) & t \in (r, s] \end{cases}$

Given this intricate way of streaming experts, the size of the active set of experts at time $t$ is at most $\log t + 1$ [12, 4]). Then, for some interval size $k$, algorithm 7 guarantees strongly adaptive regret of at most

$$\text{SA-Regret}_T^A(k) \leq O(\sqrt{k} \ln T)$$

### Computational Complexity

The strongly adaptive learning algorithm (SAOL) 7 does much of the same as the FLH algorithm, but instantiates only $\log t$ base learners at time point $t$. This

reduces the computational overhead *at time point* $t$ to $O(\log t)$. Again given observations $\{\mathbf{x}_i\}^T \in \mathbb{R}^d$, each algorithm can compute gradients and update its weight vectors in $O(d)$ time. No additional computational complexity is added by learning via instantaneous regret rather than loss. As such, the time complexity of SAOL is $O(dT \log T)$.

### 5.4 No Need for Feasible Set Defence

Both the adaptive and strongly adaptive online learning algorithms introduced in the previous subsections in their analyses assume bounded losses, and as such cannot tolerate unbounded outliers. Also for many other algorithms, this traditionally necessitates the implementation of a feasible set defence [24], i.e. a projection of each data point onto a *feasible set* with bounded diameter. This is to prevent trivial attacks that change data s.t. the norm of the resulting gradients will be large, resulting in large changes of the parameters.

Looking more closely, however, we observe that for all the algorithms mentioned we require the boundedness assumptions only to cover the interval(s) of interest, i.e. those containing only good data points.

The adaptive and strongly adaptive online learning algorithms thus do not require the loss on the outlier data points to be bounded to achieve low robust regret. Rather, the feasible set defence is made redundant by algorithms 5, 6 and 7.

We observe in the proof to theorem 5.1 that *nowhere* does the proof rely on the boundedness of the losses incurred *outside* of the interval $I = [r, s]$. This obviously implies that the loss of the outliers can be unbounded (but must be non-negative).

We discover that this in fact holds also for the two algorithms that we introduced afterwards: The adaptive regret algorithm 6 and the strongly adaptive regret algorithm 7.

For algorithm 6, the proof to its regret bound uses relative entropy (as does the proof to the slightly improved bound for the same algorithm, in appendix B. At no step in the proof does the regret bound of the interval $I = [r, s]$ require losses incurred outside of that interval to be bounded.

The same applies to algorithm 7, whose regret bound is proven using a potential function. Again, throughout the proof, the regret bound requires losses to be bounded only in between $r$ and $s$.

In the following chapter, we implement the algorithms described in the previous sections and compare their robust regret to that of the base learner for a varying fraction of adversarial outliers.

# 6 Practical Implementations

To compare the algorithms mentioned in the previous chapter in terms of performance in settings with adversarial outliers, we implement them using online gradient descent with squared error loss function $\mathcal{L}_t^{(i)} = (\hat{y}_t^{(i)} - y_t)^2$ and step size $\eta_t = \frac{1}{\sqrt{t}}$, $t \in \{1, \dots, T\}$ as base learner. We choose for the hypothesis class $\mathcal{H}$ the set of linear hypotheses, s.t. for all base learners, $\hat{y}_t^T = \mathbf{x}_t \mathbf{h}$. We then compare the resulting robust regret to the performance of this base learner.

Given that the adaptive and strongly adaptive algorithms mentioned in the previous chapter are robust to the attacks mentioned in [24], we focus on simple attacks that are usually prevented via a feasible set defence, (i.e. a projection of samples onto a feasible set with bounded diameter). Since the adaptive and strongly adaptive algorithms do not require outliers to be bounded, we simulate outliers s.t. the gradients used for the parameter updates will be large.

We sample $T = 1500$ data points from $(y, x) \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}\right)$. Given that the loss is unbounded, we cap it at 1 for the *good* data points, but leave it uncapped for the outliers. The outliers are generated by replacing $\epsilon T$ randomly selected data points. Since we want to test the algorithms' performances for increasing $\epsilon$, we start by inserting 1 outlier and measuring the algorithms' total regret. Then, with the previous outlier(s) remaining in the data set, more outliers are inserted and the regret measured until there are $0.5T$ outliers in the data set.

The outliers to insert at the randomly selected indices are generated via $y \sim \text{Exp}\big(\lambda = 0.2\big), x \sim \mathcal{N}\big(25, 1\big)$.

Since the loss is capped for the good data points, the loss function would no longer be convex. To remedy this, we revert to *sampling from the set of active experts* for each algorithm, instead of taking a convex combination of their predictions. I.e., at each time point $t$, we predict by sampling experts with probabilities respective to their weights, $p_t^{(i)} = \frac{w_t^{(i)}}{\sum_{u=1}^{t} w_t^{(u)}}$.

Figure 2 displays the performance of the respective algorithms for increasing $\epsilon$, $\epsilon \in [0, 0.5]$. We observe that the non-adaptive base learner can break down with just a single outlier. In fact, the robust regret of the base learner quickly approaches infinity. As expected, all adaptive meta algorithms achieve low robust regret, even in a high-$\epsilon$ setting. Further, we observe that, as expected, the strongly adaptive learning algorithm 7 achieves lower regret than the two adaptive algorithms for every number of outliers. This is partly due to the step size. Since the intervals each individual base learner in algorithm 7 will cover are predefined, each base learner can also have an individual step size based on the interval size. Meanwhile, to obtain similar (strongly adaptive) performance, for the adaptive algorithms one would have to "guess" the step size such that

it is close to the actual interval size.

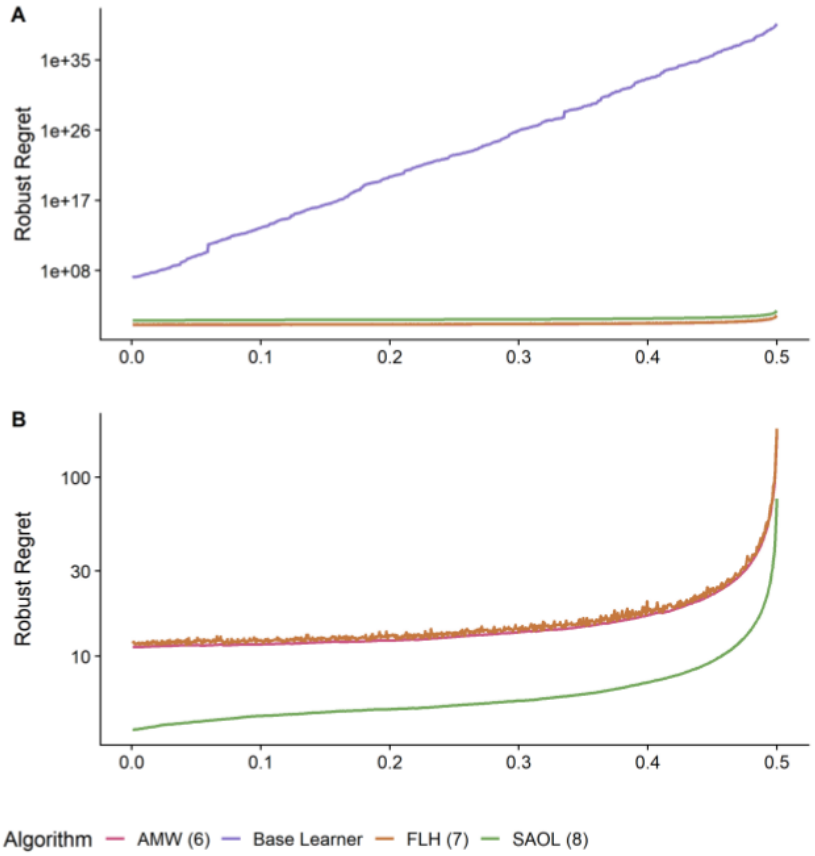



Figure 2: Performance of algorithms 5 (Adaptive Multiplicative Weights, AMW, with $\eta = 1/\log T$), 6 (Follow The Leading History, FLH), 7 (Strongly Adaptive Online Learner, SAOL) as well as the base learner (online gradient descent) on the aforementioned learning task in terms of robust regret.
**A**: All four algorithms.
**B**: Algorithms 5, 6 and 7.
The result is the average of 25 repetitions and is normalized by the square root of the number of good data points left.

# 7 Conclusion

We reviewed relevant literature on robust estimation in offline learning and discussed why it does not apply to adversarial online learning. Further, we showed that being robust to adversarial $\epsilon$-contamination following definition 1.6 is possible in the online learning setting. While robust algorithms for stochastic optimization do not quite fit into the online learning setting due to differences in assumptions and requiring batches of data, robustness can be achieved by using algorithms that are already inherently online.

Given the definition of robust regret as regret measured only on the "good" data points, adaptive algorithms that can quickly adapt to changing environments are an obvious fit.

We showed that both adaptive algorithms 6 and strongly adaptive algorithms 7 can make for robust online learning, where the strongly adaptive algorithm can handle an $o(T)$ amount of outliers.

The algorithms are robust to both of the introduced contamination scenarios: at time point $t$ having a possible $\epsilon t$ or $\epsilon T$ number of outliers. We further showed that, while the algorithms' analyses assumed bounded losses, we in fact only require the losses on the *good* data points to be bounded.

Our practical implementation shows that the strongly adaptive algorithm significantly outperforms the base learner as well as both adaptive algorithms in the face of outliers. This highlights the importance of selecting the correct step size for each base learner, as well as keeping the set of active experts small. Further research could focus on whether robustness can be guaranteed in settings where the loss of the outliers as well as the "good" data points is unbounded.

## References


[1] Amir-Hossein Bateni and Arnak S Dalalyan. *Minimax rates in outlier-robust estimation of discrete models*. Tech. rep. arXiv: 1902.04650v1.

[2] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. *Learning from Untrusted Data*. Tech. rep. 2017. arXiv: 1611.02315v2.

[3] Yu Cheng, Ilias Diakonikolas, and Rong Ge. *High-Dimensional Robust Mean Estimation in Nearly-Linear Time*. Tech. rep. 2017.

[4] Amit Daniely, Alon Gonen, and Shai Shalev-Shwartz. *Strongly Adaptive Online Learning*. Tech. rep. 2015. arXiv: 1502.07073v3.

[5] Ilias Diakonikolas et al. "Being Robust (in High Dimensions) Can Be Practical". In: (Mar. 2017). arXiv: 1703.00893.

[6] Ilias Diakonikolas et al. "Robust Estimators in High Dimensions without the Computational Intractability". In: (Apr. 2016). arXiv: 1604.06443.

[7] Ilias Diakonikolas et al. "Sever: A Robust Meta-Algorithm for Stochastic Optimization". In: (Mar. 2018). arXiv: 1803.02815.

[8] David L. Donoho and Miriam Gasko. *Breakdown Properties of Location Estimates Based on Halfspace Depth and Projected Outlyingness*. DOI: 10.2307/2242368.

[9] Jiashi Feng, Huan Xu, and Shie Mannor. "Outlier Robust Online Learning". In: (Jan. 2017). arXiv: 1701.00251.

[10] Elad Hazan. "Introduction to Online Convex Optimization". In: *Foundations and Trend in Machine Learning* 2.3-4 (2016), pp. 157–325.

[11] Elad Hazan and C. Seshadhri. *Adaptive Algorithms for Online Optimization*. 2007.

[12] Elad Hazan and C Seshadhri. "Efficient learning algorithms for changing environments". In: *ICML '09 Proceedings of the 26th Annual International Conference on Machine Learning* (2009).

[13] Peter J. Huber. "Robustness: Where are we now?" In: *Lecture Notes-Monograph Series* 31.2 (1997), pp. 487–498.

[14] Peter J. Huber and Elvezio Ronchetti. *Robust statistics*. Wiley, 2009, p. 354. ISBN: 9780470129906.

[15] Kevin A. Lai, Anup B. Rao, and Santosh Vempala. "Agnostic Estimation of Mean and Covariance". In: (Apr. 2016). arXiv: 1604.06968.

[16] Guillaume Lecué and Matthieu Lerasle. "Robust machine learning by median-of-means : theory and practice". In: (Nov. 2017). arXiv: 1711.10306.

[17] Thodoris Lykouris, Vahab Mirrokni, and Renato Paes Leme. *Stochastic bandits robust to adversarial corruptions*. Tech. rep. 2018. arXiv: 1803.09353v1.

[18] Yishay Mansour, Aviad Rubinstein, and Moshe Tennenholtz. "Robust Probabilistic Inference". In: *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*. Philadelphia, PA: Society for Industrial and Applied Mathematics, Oct. 2015, pp. 449–460. ISBN: 978-1-61197-374-7. DOI: 10.1137/1.9781611973730.31.

[19] Mingda Qiao and Gregory Valiant. "Learning Discrete Distributions from Untrusted Batches". In: (Nov. 2017). arXiv: 1711.08113.

[20] Shai Shalev-Shwartz. *Online Learning and Online Convex Optimization*. Vol. 4. 2. 2011, pp. 107–194. DOI: 10.1561/2200000018.

[21] Andrew F. Siegel. "Robust regression using repeated medians". In: *Biometrika* 69.1 (1982), pp. 242–244. DOI: 10.1093/biomet/69.1.242.

[22] Jacob Steinhardt. *Robust Learning: Information Theory And Algorithms*. Tech. rep. 2018.

[23] Jacob Steinhardt, Moses Charikar, and Gregory Valiant. "Resilience: A Criterion for Learning in the Presence of Arbitrary Outliers". In: (Mar. 2017). arXiv: 1703.04940.

[24] Yizhen Wang and Kamalika Chaudhuri. *Data Poisoning Attacks against Online Learning*. Tech. rep. 2018. arXiv: 1808.08994v1.

[25] Xuezhou Zhang and Xiaojin Zhu. *Online Data Poisoning Attacks*. Tech. rep. arXiv: 1903.01666v1.

# A Robust Regret Bound for n-Robust Multiplicative Weights Algorithm

Here we prove theorem 4.1 for achieving robust regret at most $\eta^{-1}\ln T + \eta T + \eta^{-1} + \eta^{-1}\lceil \epsilon T\rceil \ln T + R(T)$ with algorithm 4. As before, we assume that the loss is bounded, $\mathcal{L}_t^{(i)} \in [0,1]$, and $\eta \in [0, \frac{1}{2}]$.

*Proof.* For the proof we follow the common multiplicative weights proof using a potential function, with slight adaptations. We denote by $w_i^{(t)}$ the weight of the $i$th expert at time point $t$ and use the potential function $W(t) = \sum_{i=1}^{t} w_i^t$. Then we relate the sum of the weights at time point $T$, i.e. the end of the data stream, to the performance of the best expert (the one who leaves out all the corrupted data points).

As before we denote the loss of the full algorithm at time point $t$ with $\mathcal{L}_t$ and also, as in the algorithm, we have that $S = \{A \subseteq \{1,\dots,T\} \mid |A| = \lceil \epsilon T\rceil\}$, the set of all combinations of data points to leave out.

We note that for any $t$ we have that

$$\begin{aligned}
W(t+1) &= \sum_{i=1}^{t+1} w_t^{(i)} = 1 + \sum_{i=1}^{t} w_t^{(i)}(1 + \eta(\mathcal{L}_t - \mathcal{L}_t^{(i)})) \\
&\le 1 + \sum_{i=1}^{t} w_t^{(i)} \left(1 + \eta\left(\sum_{j=1}^{t} \frac{w_t^{(j)}\mathcal{L}_t^{(j)}}{\sum_{j=1}^{t} w_t^{(j)}} - \mathcal{L}_t^{(i)}\right)\right) \quad \text{(by Jensen's inequality)} \\
&= 1 + W(t) + \eta\Big(\sum_{j=1}^{t} w_t^{(j)}\mathcal{L}_t^{(j)} - \sum_{i=1}^{t} w_t^{(i)}\mathcal{L}_t^{(i)}\Big) \\
&= 1 + W(t)
\end{aligned}$$

In particular, by induction, we have that $W(T) \le T$.

We then consider the loss of the best expert. The best expert is the one who learns only on the good data points, i.e. if we have a set of outlier indices $u$, then the best expert is $E^{(j)} \mid S_j = u$.

We then also note that $\mathcal{I} = \{1,\dots,T\} \setminus S_j$, i.e. the set of good data points is the set that the best expert learns from.

Considering the resulting instantaneous regret of said expert, we know that the sum of the experts' weights must be at least

$$
\begin{aligned}
W(T) &\geq \frac{1}{\binom{T}{\lceil \epsilon T \rceil}} \prod_{t \in \mathcal{I}} (1 + \eta \mathcal{R}_t^{(j)}) \\
\ln W(T) &\geq \sum_{t \in \mathcal{I}} \ln(1 + \eta \mathcal{R}_t^{(j)}) - \ln \binom{T}{\lceil \epsilon T \rceil} \\
&\geq \sum_{t \in \mathcal{I}} \eta \mathcal{R}_t^{(j)} - \sum_{t \in \mathcal{I}} (\eta \mathcal{R}_t^{(j)})^2 - \ln \binom{T}{\lceil \epsilon T \rceil} \quad \text{since } \ln(1+x) \geq x - x^2 \text{ for } x \in [-\frac{1}{2}, \frac{1}{2}] \\
&\geq \eta (\sum_{t \in \mathcal{I}} \mathcal{R}_t^{(j)} - \eta T) - \ln \binom{T}{\lceil \epsilon T \rceil}
\end{aligned}
$$

As such,

$$
\begin{aligned}
\eta (\sum_{t \in \mathcal{I}} \mathcal{R}_t^{(j)} - \eta T) - \ln \binom{T}{\lceil \epsilon T \rceil} &\leq \ln T \\
\sum_{t \in \mathcal{I}} \mathcal{R}_t^{(j)} &\leq \eta^{-1} \ln T + \eta T + \eta^{-1} \ln \binom{T}{\lceil \epsilon T \rceil} \\
&\leq \eta^{-1} \ln T + \eta T + \eta^{-1} \ln \left( \frac{eT}{\lceil \epsilon T \rceil} \right)^{\lceil \epsilon T \rceil} \\
&\leq \eta^{-1} \ln T + \eta T + \eta^{-1} \lceil \epsilon T \rceil \left( \ln eT - \ln \lceil \epsilon T \rceil \right) \\
&\leq \eta^{-1} \ln T + \eta T + \eta^{-1} \lceil \epsilon T \rceil + \eta^{-1} \lceil \epsilon T \rceil \ln T - \eta^{-1} \lceil \epsilon T \rceil \ln \lceil \epsilon T \rceil \\
&\leq \eta^{-1} \ln T + \eta T + \eta^{-1} + \eta^{-1} \lceil \epsilon T \rceil \ln T
\end{aligned}
$$

Combining this with the regret of the base learner completes the proof. $\square$

# B Flexible Adaptive Regret Bound for FLH with Variable Learning Rate

Here we prove theorem 5.2 for achieving regret of $O(\sqrt{|I|}\ln|I|) + R(|I|)$ for some interval $I = [r, s]$. Our worst-case adaptive regret is then $O(\sqrt{T}\ln T)$ just as in [11], but is more useful for analyzing intervals of "good" data points since the regret no longer depends on the position of the interval in the data, but rather just the size of the interval.

We first restate lemma 28 from [11] which is needed for the proof of theorem 5.2. For ease of notation, we revert to the notation introduced in the online learning framework, i.e. the loss is denoted $f_t(\mathbf{h}_t)$.

As before, we assume that our loss is bounded, i.e. $f_t(\mathbf{h}) \in [0, M]$.

**Lemma B.1.** ([28] in [11])

1. For any $i < t$: $f_t(\mathbf{h}_t) - f_t(\mathbf{h}_t^{(i)}) \leq \eta_t^{-1}(\ln \hat{p}_{t+1}^{(i)} - \ln \hat{p}_t^{(i)} + \eta_t^2 M^2 + 2/t)$
2. $f_t(\mathbf{h}_t) - f_t(\mathbf{h}_t^{(t)}) \leq \eta_t^{-1}(\ln \hat{p}_{(t+1)}^{(t)} + \eta_t^2 M^2 + \ln t)$

*Proof.*

$$\begin{aligned}\ln p_t^{(i)} - \ln \hat{p}_{t+1}^{(i)} &= \ln p_t^{(i)} - \ln \frac{p_t^{(i)} e^{-\eta_t f_t(x_t^i)}}{\sum_{j=1}^t p_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})}} \\ &= \eta_t f_t(\mathbf{h}_t^{(i)}) + \ln(\sum_{j=1}^t p_t^{(j)} e^{-\eta_t f_t(\mathbf{h}_t^{(j)})})\end{aligned}$$

$$\begin{aligned}\sum_{\ell=1}^t p_t^{(\ell)} p_t^{(\ell)} \ln \frac{\hat{p}_{t+1}^{(\ell)}}{p_t^{(\ell)}} &= \sum_{\ell=1}^t p_t^{(\ell)} \ln \frac{e^{-\eta_t f_t(\mathbf{h}_t^{(\ell)})}}{\sum_{j=1}^t p_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})}} \\ &= -\eta_t \sum_{\ell=1}^t p_t^{(\ell)} f_t(\mathbf{h}_t^{(\ell)}) - \ln(\sum_{j=1}^t p_t^{(j)} e^{-\eta_t f_t(\mathbf{h}_t^{(j)})})\end{aligned}$$

Then, using the convexity of $f_t$ and putting the above equations together:

$$\begin{aligned}f_t(\mathbf{h}_t) - f_t(\mathbf{h}_t^{(i)}) &= f_t(\sum_{\ell=1}^t p_t^{(\ell)} x_t^{(\ell)}) - f_t(\mathbf{h}_t^{(i)}) \\ &\leq \sum_{\ell=1}^t p_t^{(\ell)} f_t(\mathbf{h}_t^{(\ell)}) - f_t(\mathbf{h}_t^{(i)}) \\ &= \eta_t^{-1}\big(\ln \hat{p}_{t+1}^{(i)} - \ln p_t^{(i)} - \sum_{\ell=1}^t p_t^{(\ell)} \ln \frac{\hat{p}_{t+1}^{(\ell)}}{p_t^{(\ell)}}\big)\end{aligned}$$

$$
\begin{aligned}
-\sum_{\ell=1}^{t} p_t^{(\ell)} \ln \frac{\hat{p}_{t+1}^{(\ell)}}{p_t^{(\ell)}} &= \eta_t \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(\mathbf{h}_t^{(\ell)}) + \ln(\sum_{\ell=1}^{t} p_t^{(\ell)} e^{-\eta_t f_t(\mathbf{h}_t^{(\ell)})}) \\
&= \eta_t \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(\mathbf{h}_t^{(\ell)}) + \ln(1 - (1 - \sum_{\ell=1}^{t} p_t^{(\ell)} e^{-\eta_t f_t(x_t^{(\ell)})})) \\
&\leq \eta_t \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(\mathbf{h}_t^{(\ell)}) - 1 + \sum_{\ell=1}^{t} p_t^{(\ell)} e^{-\eta_t f_t(x_t^{(\ell)})} \\
&\leq \eta_t \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(x_t^{(\ell)}) - 1 + \sum_{\ell=1}^{t} p_t^{(\ell)} (1 - \eta_t f_t(\mathbf{h}_t^{(\ell)}) + (\eta_t f_t(x_t^{(\ell)}))^2) \\
&= \eta_t^2 \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(\mathbf{h}_t^{(\ell)})^2
\end{aligned}
$$

As such,

$$
f_t(\mathbf{h}_t) - f_t(\mathbf{h}_t^{(i)}) \leq \eta_t^{-1} \big( \ln \hat{p}_{t+1}^{(i)} - \ln p_t^{(i)} + \eta_t^2 \sum_{\ell=1}^{t} p_t^{(\ell)} f_t(\mathbf{h}_t^{(\ell)})^2 \big)
$$

□

We further use claim 11 from [11] stating that $\ln p_t^{(i)} \geq \ln \hat{p}_t^{(i)} - 2/t$ and $\ln p_t^{(t)} \geq -t \ln t$. With this in hand, we can prove theorem 5.2. We follow the proof to lemma 27 in [11] but bound the terms differently.

*Proof.* Unlike in Hazan and Seshadhri, we set the first term $f_r(x_r) - f_r(x_r^{(r)}) = \max f_r(x_r) - f_r(x_r^{(r)}) = M$. Then,

$$
\begin{aligned}
\sum_{t=r}^{s} \big( f_t(\mathbf{h}_t) - f_t(\mathbf{h}_t^{(r)}) \big) &= \big( f_r(x_r) - f_r(x_r^{(r)}) \big) + \sum_{t=r+1}^{s} \big( f_t(\mathbf{h}_t) - f_t(\mathbf{h}_t^{(r)}) \big) \\
&\leq M + \sum_{t=r+1}^{s} \eta_t^{-1} \big( \ln \hat{p}_{t+1}^{(r)} - \ln \hat{p}_t^{(r)} + \eta_t^2 M^2 + 2/t \big) \\
&= M + \Big( \sum_{t=r+2}^{s+1} (\eta_{t-1}^{-1} - \eta_t^{-1}) \ln \hat{p}_t^{(r)} \Big) - \eta_{r+1}^{-1} \ln \hat{p}_{r+1}^{(r)} + \eta_{s+1}^{-1} \ln \hat{p}_{s+1}^{(r)} + \\
&\quad \sum_{t=r+1}^{s} \eta_t^{-1} \big( \eta_t^2 M^2 + 2/t \big) \\
&\leq M - \eta_{r+1}^{-1} \ln \hat{p}_{r+1}^{(r)} + \Big( \sum_{t=r+2}^{s+1} (\eta_{t-1}^{-1} - \eta_t^{-1}) \ln \hat{p}_t^{(r)} \Big) + \sum_{t=r+1}^{s} M^2 \eta_t + \sum_{t=r+1}^{s} 2/(t\eta_t)
\end{aligned}
$$

From the definition of $\hat{p}_t^{(r)}$ we also have that

$$\begin{aligned}
-\eta_{r+1}^{-1} \ln \hat{p}_{r+1}^{(r)} &= \eta_{r+1}^{-1} \ln \left( \frac{\frac{1}{r} e^{-\eta_r f_r(x_r^{(r)})}}{\sum_{j=1}^{r} p_r^{(j)} e^{-\eta_r f_r(x_r^{(j)})}} \right) \\
&= -\eta_{r+1}^{-1} \left( -\ln r - \eta_r f_r(x_r^{(r)}) - \ln \sum_{j=1}^{r} p_r^{(j)} e^{-\eta_r f_r(x_r^{(j)})} \right) \\
&\leq \eta_{r+1}^{-1} \ln r + \frac{\eta_r}{\eta_{r+1}^{-1}} f_r(x_r^{(r)})
\end{aligned}$$

Setting the learning rate $\eta_t = 1/\sqrt{t}$,

$$-\eta_{r+1}^{-1} \ln \hat{p}_{r+1}^{(r)} \leq \sqrt{r+1} \ln r + \frac{\sqrt{r+1}}{\sqrt{r}} M$$

And therefore, returning to summing up the difference between the loss of the algorithm and the best expert, we obtain

$$\begin{aligned}
\sum_{t=r}^{s} \big( f_t(\mathbf{h}_t) - f_t(\mathbf{h}_t^{(r)}) \big) &\leq M + \sqrt{r+1} \ln r + \sqrt{2} M - \sum_{t=r+2}^{s+1} \ln \hat{p}_t^{(r)} (\sqrt{t} - \sqrt{t-1}) + \\
&\qquad M^2 \sum_{t=r+1}^{s} 1/\sqrt{t} + 2 \sum_{t=r+1}^{s} \sqrt{t}/t \\
&\leq M + \sqrt{2} M + \sqrt{r+1} \ln r - \sum_{t=r+2}^{s+1} \ln \hat{p}_t^{(r)} (\sqrt{t} - \sqrt{t-1}) + \\
&\qquad M^2 \sum_{t=1}^{s-r} 1/\sqrt{t} + 2 \sum_{t=1}^{s-r} 1/\sqrt{t} \\
&\leq (\sqrt{2}+1) M + \sqrt{r+1} \ln r - \sum_{t=r+2}^{s+1} \ln \hat{p}_t^{(r)} (\sqrt{t} - \sqrt{t-1}) + (2M^2 + 4) \sqrt{|I|}
\end{aligned}$$

Using Hazan and Seshadhri's result that $\hat{p}_t^{(r)} \geq O(t^{-1})$,

$$\begin{aligned}
\sum_{t=r}^{s} \big( f_t(\mathbf{h}_t) - f_t(\mathbf{h}_t^{(r)}) \big) &\leq (\sqrt{2}+1) M + \sqrt{r+1} \ln r - \sum_{t=r+2}^{s+1} \ln \hat{p}_t^{(r)} (\sqrt{t} - \sqrt{t-1}) + (2M^2 + 4) \sqrt{|I|} \\
&\leq (\sqrt{2}+1) M + \sqrt{r+1} \ln r + \sum_{t=r+2}^{s+1} - \ln O(t^{-1}) (\sqrt{t} - \sqrt{t-1}) + (2M^2 + 4) \sqrt{|I|} \\
&\leq (\sqrt{2}+1) M + \sqrt{r+1} \ln r + \sum_{t=r+2}^{s+1} O\big( (\ln t) \big( \sqrt{t} - \sqrt{t-1} \big) \big) + (2M^2 + 4) \sqrt{|I|} \\
&\leq (\sqrt{2}+1) M + \sqrt{r+1} \ln r + O\big( \sqrt{|I|} \ln |I| \big) + (2M^2 + 4) \sqrt{|I|} \\
&\leq O\big( \sqrt{|I|} \ln |I| + \sqrt{r} \ln r \big)
\end{aligned}$$

□

The regret for interval $I$ is thus upper bounded by $R(|I|) + O\big(\sqrt{|I|}\ln|I| + \sqrt{r}\ln r\big)$, where $R(|I|)$ is the regret of the best expert $E^{(r)}$.

# Tiedemann Thesis

GRADEMARK RAPPORT

EINDCIJFER

/0

ALGEMENE OPMERKINGEN

**Docent**

PAGINA 1

PAGINA 2

PAGINA 3

PAGINA 4

PAGINA 5

PAGINA 6

PAGINA 7

PAGINA 8

PAGINA 9

PAGINA 10

PAGINA 11

PAGINA 12

PAGINA 13

PAGINA 14

PAGINA 15

PAGINA 16

PAGINA 17

PAGINA 18

PAGINA 19

PAGINA 20

PAGINA 21

PAGINA 22

PAGINA 23

PAGINA 24

PAGINA 25

PAGINA 26

PAGINA 27

PAGINA 28

PAGINA 29

PAGINA 30

PAGINA 31

PAGINA 32

PAGINA 33

PAGINA 34

PAGINA 35

PAGINA 36

PAGINA 37

PAGINA 38

PAGINA 39

PAGINA 40