universitätbonn

Fraunhofer IAIS

Rheinische Friedrich-Wilhelms-Universität Bonn

Mathematisch-naturwissenschaftliche fakultät

Master thesis

Online Robust Learning Using Radon Point

Author:
Rania Briq

First Examiner:
Prof. Dr. Stefan Wrobel

Second Examiner:
PD. Dr. Michaek Mock

Submitted: 15.05.2017

Declaration of Authorship

I, Rania Briq, confirm that this work is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g. ideas, equations, figures, text, tables, programs) are properly acknowledged at the point of their use. A full list of the references employed has been included.

Abstract

This thesis analyzes a novel approach for model synchronization in distributed online learning from noisy data streams. The proposed approach combines weak hypotheses that have been computed locally by replacing them with their Radon point. These hypotheses may be learned by a wide range of online learning algorithms thus making the approach black-box.

The work encompasses both the theoretical and empirical aspects of the method. The theoretical analysis focuses on proving probabilistic guarantees on the error bound. We show that on noise-free streams, the approach satisfies strong probabilistic error guarantees within the framework of PAC (Probably Approximately Correct) learning. Additionally, under strict assumptions, it provides a method for converting regret bounds of standard online learning algorithms to PAC error bounds. The empirical part focuses on evaluating the practical aspect of the approach. It shows that the proposed approach outperforms state of the art approaches on noisy data streams.

Contents

1 Introduction **1**
- 1.0.1 Motivating the Radon Point . . . 4
- 1.1 Problem statement . . . 6

2 Related Work **7**

3 Method and Theoretical Analysis **11**
- 3.1 Overview . . . 11
- 3.2 Theoretical Background . . . 12
- 3.3 Radon Point . . . 15
  - 3.3.1 PAC learnability . . . 20
- 3.4 Method Description . . . 21
- 3.5 Theoretical Analysis . . . 23
  - 3.5.1 Theoretical Background . . . 23
  - 3.5.2 PAC for the Radon point . . . 24
  - 3.5.3 Converting Regret Bound to PAC . . . 26
  - 3.5.4 Interpretation . . . 31

4 Empirical Evaluation **39**
- 4.1 Setup . . . 39
- 4.2 Experiments . . . 40
  - 4.2.1 Evaluation on Uncontaminated data streams . . . 40
  - 4.2.2 Evaluation on noisy data streams . . . 41
  - 4.2.3 Evaluation of the Radon point versus the Geometric median 43
  - 4.2.4 Complexity and Runtimes . . . 43
  - 4.2.5 Dependence on parameters . . . 45

5 Discussion, Conclusion and Future Work **59**
- 5.1 Discussion . . . 59
- 5.2 Conclusion . . . 60
- 5.3 Future work . . . 61

6 Acknowledgements **63**

7 Appendix **65**

1 Introduction

This thesis proposes a novel approach for model synchronization in a distributed online learning environment with noisy data streams. In particular, it considers data streams that are generated by sensor nodes which occasionally break down and recover later after a period of time.

Data streams have become prevalent in a wide range of areas. For instance, the widespread use of sensor networks for accomplishing various tasks has turned sensor networks into an important source of data streams [26, 4].

A sensor network is a group of small sensor devices that perform a task collaboratively. A single sensor measures a single value or a small set of values repeatedly. Areas of application for sensor networks include monitoring, tracking and controlling.

These data streams have the potential of being analyzed and used for real-time prediction, which can be necessary for a sensor network's designated task, such as object tracking, nuclear reactor controlling and fault detection [13].

The technology in sensors is improving due to Moore's law, however, the cost of communication remains stagnant [13]. The increasing computation power makes it possible to push computations towards the sensor layer rather than sending the data to a central node. Sending the data to a central node over the network and waiting for a response imposes delay in making predictions due to network latency, thereby making it hard to realize real-time prediction. On the other hand, sensors are still small devices limited in their power, therefore assigning to them a compact model that can be learned with a small amount of computations makes them capable of making real-time predictions. This can be achieved by an online learning algorithm, which upon an incoming sample updates the model with a few simple steps and then discards the sample.

Additionally, sensor nodes are prone to failure especially in harsh environments [12, 8, 26, 4], resulting in erroneous readings and contamination in their data streams. Online learning algorithms need to address this noise, otherwise, they might fail in producing good models that can produce accurate predictions.

The goal of this thesis is to model the scenario described above, which includes multiple noisy data streams for which real-time prediction needs to be made, into a distributed online learning system.

Generally, the task of a machine learning algorithm can be defined as finding an approximate hypothesis that is as close to the optimal hypothesis as possible. In a general learning setup, the data is generated by some unknown distribution over a domain set of an input and output space and the goal is to find the best model for that distribution that realizes this hypothesis.

The general setting considered in this thesis consists of multiple data streams and learners, one for each stream. The goal of each learner is to predict the labels of incoming instances as they arrive and learn the corresponding unknown target function collaboratively with the other learners. To perform this task, each learner maintains a local model that it updates upon seeing a new sample and periodically synchronizes this model with the rest of the learners. Due to memory and space constraints, the sample is discarded as soon as it is processed and cannot be accessed again.

The approach proposed herein uses a linear model algorithm, that is, the model assumes a linear relationship between the input and the output whose value it predicts. The model is represented by a $d+1$ vector in the Euclidean space, $d \in \mathbb{N}$ denotes the data dimensionality. Running multiple instances of an online learning algorithm on distributed nodes allows handling multiple streams. However, since the data streams are generated by the same distribution, in complete absence of communication between the nodes, the performance would not be optimal. Since an optimal solution is obtained by a hypothetical super central node which observes all the data, the performance can be improved by enabling an exchange of information between the learners. The form of information exchange used in the proposed scheme relies on the learned model, which is linear in the size of the data dimension. Such a model is a compressed representation of the observed data and when communicated periodically the amount of data exchanged remains bounded compared to the number of observed samples. Synchronizing the models therefore guarantees a better representative hypothesis and accordingly an enhanced prediction quality.

The synchronization scheme proposed for this setting comprises computing an aggregated global model based on all local models and substituting this model back at each learner. To that end, it relies on an additional node that we refer to as the Coordinator. The learners send their models after seeing a predefined number of samples $b \in \mathbb{N}$ to the coordinator, which in turn combines the models and sends the result back to the learners. Every learner then saves this model and continues to learn and update it locally until the next synchronization phase.

This work addresses the method that the coordinator uses to combine the hypotheses. Tackling the problem of learning from decentralized noisy data streams requires designing robust online distributed learning algorithms. Online learning

methods which do not take into account the existence of noise in the data may break down if the data is noisy. For instance, aggregating by averaging [9] is attractive due to its simplicity and speed but in case of noisy data at a certain node, the model learned will not reflect well on the true distribution. Since averaging is inherently susceptible to outliers, such a bad model can deteriorate the aggregated model substantially when performing synchronization.

Another existing solution for online distributed data streams has already been proposed in Kamp et al. [19], their solution is communication efficient but since it employs averaging for combining the models, it might not work when the data streams are noisy. A solution for noisy data streams which employs the geometric median has been proposed in Feng et al. [11], which is a robust approach for noisy data streams. However, the method still has shortcomings which are demonstrated in its computation time and its susceptibility to unbounded outliers as will be shown empirically later.

We aim to find a robust approach that can prevent or attenuate the distortion of the global hypothesis and then evaluate its performance. The proposed approach in this thesis uses the Radon point of the learners' hypotheses as an aggregated global hypothesis. Using this point is motivated by its central property which states that the Radon point of a given set is better than at least two points in this set with respect to some quality measure.

The approach builds upon the algorithm in Kamp et al. [20] which is suited for offline batch learning. The setting to which they apply their algorithm consists of several distributed learners. Each of these learners is assigned a subset of dataset which it processes as a batch to construct its own independent weak hypothesis. The "weakness" of a local hypothesis is attributed to the fact that a single learner does not observe all the data arriving from the distribution.
After each learner has observed all the samples in its assigned batch and built its own hypothesis, all the local hypotheses are aggregated iteratively into one hypothesis using the Iterated Radon Points algorithm [7]. In their approach, the strong hypothesis is the Radon point of all the local models. The authors further show a probabilistic bound on the solution quality of the aggregated model, which is doubly exponentially stronger than the bound obtained for a single weak hypothesis at a base learner.

Their method, whose guarantees hold for a batch learning mode, is applied in this work to a distributed online learning setting in a mini-batching manner. In particular, the PAC guarantees obtained in their offline batch mode are used as a base guarantee and then extended in subsequent synchronization rounds, thereby providing further guarantees for the online case with diminishing error probability.

1.0.1 Motivating the Radon Point

The Radon point presents itself as a good candidate for an aggregation operator owing to several of its properties. The first fundamental property, which is also used to establish the probabilistic guarantees of the approach, lies in the fact that the Radon point of a set is better than at least two points in this set. This property does not hold for the average point for example. It will be formalized later on and for now we briefly mention that this property constitutes the basis of a theoretical error bound, which the average point fails to provide.

Second, as opposed to averaging, the Radon point is not as susceptible to outliers. The justification for this claim can be established from a chain of related definitions. While these definitions don't constitute a basis for the proof on the probabilistic guarantees obtained, they provide an indication as to why the Radon point is a viable candidate.
We start from the paper Clarkson et al. [7], which includes the main algorithm whose output is used in the proposed approach as the aggregated model. In this paper, it is argued that the point calculated by the algorithm is an approximation of the center point. A center point of a set of points partitions the set into two balanced subsets. A formal definition of the center point will be given later. This balanced partitioning that is induced by the center point already indicates the robustness of Radon point against outliers and its improbability for being easily drawn far from the proximity of the majority of the models points. This is the first indication to the potential performance of the Radon point. Note that throughout the paper, we refer to the output of the algorithm in Clarkson et al. [7] as Radon point when we actually mean the point produced by the Iterated Radon Points algorithm.

The second indication is based on Donoho and Gasko [10], which suggests center points as estimators for multivariate datasets. In this paper, they rely on the notion of depth of a point, also known as the Tukey depth. For a one-dimensional dataset S, the Tukey depth of a point x is defined to be the minimum between the number of the points on the left and right of x For a d-dimensional set, the depth of $x \in \mathbb{R}^d$ is defined as the minimum depth of any one-dimensional projection of the dataset. When $d = 1$, the deepest point is the median, and in higher dimensions, the deepest point may be considered as a multi-dimensional median as well. This also reinstates the robustness of points whose depth is large such as center points. These points have a high breakdown point, where the breakdown point can be defined as the minimum number of outliers required to break down an estimator represented by such a point [10]. For instance, one bad point is sufficient to break down the average whereas it takes approximately half the points of a set to break

down the median in a one-dimensional space.

The third indication is based on Gilad-Bachrach et al. [15]. They argue that when learning a linear classifier, the Bayes Point([16]) is almost identical to the Tukey median and center point. This indicates that a Radon point is potentially a good classifier due to the results accomplished by a Bayes Point classifier, which are comparable to those of support vector machines which are a powerful model for classification. Furthermore, the bound on the generalization error of the Bayes Point is bounded by that of the Bayes classifier. We note that the Radon point is more general than the Bayes classifier.

All these indications put together suggest that a Radon point is robust and relate it to other powerful classifiers and therefore make it a good candidate for a classifier.

Fig. 1.1 illustrates the robustness of the Radon point versus the susceptibility of the average point for a small two-dimensional set that contains one outlier.

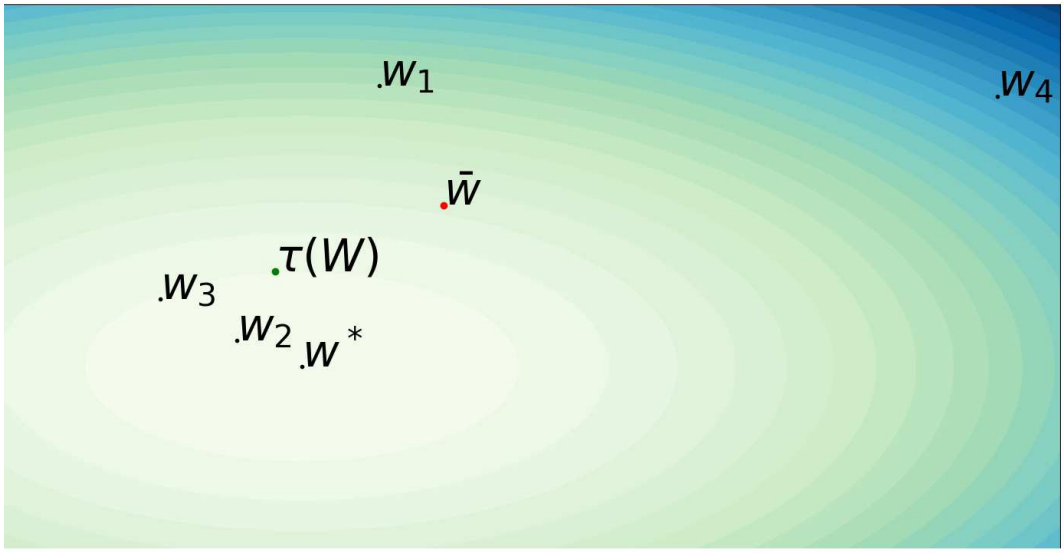


Figure 1.1: The plot shows an example of the robustness of a Radon point compared to the average point in the presence of an outlier. The set consists of 4 data points $w_1, \ldots, w_4$ in $\mathbb{R}^2$ where w_4 is an outlier. It can be seen that w_4 drags the average away from the optimum w^* whereas the Radon point $\tau(W)$ is less influenced by the outlier point and remains closer to the optimum

The proposed algorithm is black-box assuming that the underlying optimization algorithm is convex and consistent and that the hypothesis being learned is representable as a vector in a finite dimensional Euclidean space. Given that these conditions are satisfied, then the overall algorithm is agnostic to the optimization algorithm used and allows for it to be applied as is.

The approach is analyzed on a theoretical level in which error bounds are provided within the framework of PAC learning and then evaluated in practice in two scenarios: the first is when the observed samples are free of noise and the second is when the samples are occasionally contaminated by noise. The question that we wish to pursue is whether a Radon point is a good synchronization approach.

1.1 Problem statement

The problem that we tackle in this thesis is described as follows:
given $k \in \mathbb{N}$ data streams generated by the same distribution $D : X \times Y \to [0, 1]$ where $X \subseteq \mathbb{R}^d$ comprises the samples feature space and $Y \subseteq \mathbb{R}$ is the output space, k nodes each handling a single stream, an online learning algorithm A and a loss function that measures the prediction quality of A for a sample (x_t, y_t) at a given time t. Each node is running a separate instance of A and trying to learn a hypothesis that predicts the label y_t by $\hat{y}_t$ as accurately as possible. Every computation node has a failure probability $P_{failure}$ and a recovery probability $P_{recovery}$. Every feature of an instance arriving in a failure interval is perturbed by a Gaussian additive noise $N(0, \sigma)$ with probability P_{noise}, $\sigma \in \mathbb{R}$ and determines the magnitude of the noise. These perturbed samples may distort a learned hypothesis from the true hypothesis. Seeing that the samples across the different streams arrive from the same distribution, combining the local individual hypotheses may produce an enhanced hypothesis which replaces local hypotheses, thereby improving the overall prediction quality. However, combining the hypotheses in the presence of one or more corrupted hypotheses may yield a distorted global hypothesis, thereby degrading the learning quality. The objective is to study a robust aggregation approach based on Radon points that can protect against the distortion of the global hypothesis from some outlier hypotheses.

2 Related Work

In this chapter, several papers on distributed learning, online learning and learning from noisy data are discussed and their relation to the proposed approach is shown.

Dekel et al. [9] proposes a distributed online mini-batch algorithm for multiple data streams. Their synchronization method, which is based on averaging the local hypotheses, cannot cope with noisy data. Since the proposed approach is closely related to theirs, I elaborate about their method in more detail in the method description.

Kamp et al. [19] has a similar setting to mine and proposes an online distributed learning algorithm. This work focuses on solving the question of when the learners should communicate in order to synchronize their models. The communication in this algorithm is governed by a dynamic synchronization scheme within which the learners communicate only if their local models diverge from a global reference point. This is in contrast to a static synchronization scheme within which the learners communicate periodically after observing a fixed number of samples, regardless of the local conditions. This approach achieves two goals. First, when the local learners have not converged yet, the frequency of communication is higher which helps them converge faster. Second, once the learners have converged, then if there is no concept drift, the local models will communicate less frequently thereby saving in the amount of communication. Their approach also relies on averaging for combining the models and is therefore unsuitable for scenarios where the data is noisy.

The communication complexity of distributed convex optimization has been studied in Arjevani and Shamir [1] for a batch setting. Given m machines each assigned a different local convex loss function and a subset of the data set, the authors answer the question of how much communication is needed in order to optimize the average of these m local functions up to some given accuracy ϵ. The subsets are not necessarily split uniformly at random between the machines and they provide different guarantees based on different assumptions on the statistical similarity between the subsets.

In Mcdonald et al. [22], the authors present a distributed gradient computation method. In their setting, there exists p distributed machines and a sample set $S = (S_1, \ldots, S_p)$ whose data samples are i.i.d. The dataset is divided into p

subsamples $S_i, 1 \leq i \leq p$, each of size m. Each of the learner computes a local gradient in parallel. The local gradients are then summed up into a global gradient on a single machine that updates the weights vector. Their setting conforms to offline distributed batch learning.

Rosenblatt and Nadler [24] analyze the optimality of averaging in a distributed learning setting. They assume a big dataset of size $n \to \infty$ which is divided among m machines each trying to estimate its own M-estimator $\widehat{\theta}_i, i \in [m]$. They analyze the statistical properties of the averaged estimator given by $\bar{\theta} = \frac{1}{m}\sum_{i=1}^{m}\widehat{\theta}_i$ in two settings: the first is when p, the number of the parameters to be estimated, is a constant and the second is related to a high-dimensional regime in which $p \to \infty$. In either case, the setting is an offline learning scenario rather than in-stream.

Johnson and Zhang [18] argues that while stochastic gradient descent relies only on one derivative for each update step, it increases the variance in the computed derivatives which could result in slow convergence. To resolve that, they propose a method that reduces the variance. They maintain a snapshot of the estimated model as $\tilde{w}$ after some number of SGD iterations for example, additionally, they maintain the average gradient at these snapshot points. While the method is online, as can be seen, it is still relevant only to a single learner rather than a distributed group of learners. The averaging therefore is applied over time for the same learner rather than multiple distributed learners.

Online learning from noisy data has been studied in Cesa-Bianchi et al. [6] with respect to the information available on the noise distribution. They show that when such information is known, it becomes possible to learn linear and kernel-based predictors using one or two perturbed copies of a sample depending on the loss function and the given information on the noise. Such information includes for instance the mean of the noise and its finite variance bound. In the absence of any information about the distribution, however, it becomes mandatory to have multiple perturbed copies of a sample, otherwise, they show by proving a tight lower bound, that it is impossible to have a sublinear regret just from a single perturbed copy. In practice, such copies of a single samples are not necessarily available. Their setting and analysis applies to a single learner rather than a distributed set of learners.

In Feng et al. [11], the authors directly address the problem of online learning from noisy data. Their approach can be extended to a distributed setting as well. They propose a Robust Online Learning (ORL) framework which employs the geometric median as an aggregation operator. For a given set of m points $x_1 \ldots, x_m, x_i \in \mathbb{R}^n$, the geometric median x is given by:

$$x = \underset{y \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{m} ||x_i - y||$$

In other words, it is the point that minimizes the sum of distances to the points of a given set. In the one-dimensional space, this is exactly the median point. In a higher dimension, akin to the center point, the geometric median has a central tendency. They employ mini-batching and assume that within a mini-batch outliers may occur and their distribution across the mini-batches is heterogeneous. The authors further analyze the performance of their framework when the number of outliers is a constant fraction of the data. Their analysis refers to a online learning framework with a single node and they note that it can be adapted to a distributed setting in a straightforward manner. The set-up of the problem that they try to solve consists of $N = n + n_1$ data samples $\{x_1, \ldots, x_{n+n_1}\}, x_i \in \mathbb{R}^p$of which n samples reflect the true underlying distribution and n_1 are outliers. The assumption is that samples arriving from the true distribution are sub-Gaussian random variables, i.e., they satisfy

$$\mathbb{P}[|\langle x, u \rangle| > t] \leq 2e^{-\frac{t^2}{L^2}}$$

for $t > 0$, some L and $u \in S^{p-1}$ denotes the unit sphere. In their setting that consists of a single online learner, the samples arrive in T mini-batches, where the size of each mini-batch is denoted by $b = \frac{n+n_1}{T}$. They consider the problem as an online statistical learning whose goal is to estimate the parameter $\theta^* \in \Theta$ of the true distribution based on the data observed up until the point of estimation. If we denote the parameters learned from each batch by $(\theta_1 \ldots, \theta_T)$, then the geometric median is given by:

$$\widehat{\theta} = median(\theta_1, \ldots, \theta_T) := \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{j=1}^{T} |||\theta - \theta_j||.$$

The robustness of the geometric median can be derived from the following lemma:

Lemma 1 ([11]). *Let $\widehat{\theta}$ be the geometric median of the points $\theta_1 \ldots, \theta_T \in \Theta$. Fix $\gamma \in (0, 1/2)$ and $C_\gamma = (1 - \gamma)\sqrt{\frac{1}{1-2\gamma}}$. Suppose that there exists a subset $S_\theta \subseteq \{\theta_1, \ldots, \theta_T\}$ of cardinality $|S_\theta| > (1 - \gamma)T$, then for any point $\theta_j \in S_\theta$ and any point $\theta^* \in \Theta$, it holds that*

$$||\theta_j - \theta^*|| \leq C_\gamma r$$

The lemma states that the geometric median of a set will be close to any point

in the space, in particular also the optimal point, if at least half the points do not deviate by much from that point, and the distance from the optimal point then is not influenced by the outliers distance form the optimal point at all, but rather only by the fraction of the outliers and the distance of the authentic parameters.

In their algorithm, they apply a two-stage filtering. In the first stage, filtering is applied on a mini-batch level in order to prevent outliers from distorting the estimation of the local parameter θ_j. The motivation behind this step lies in having outliers with a large magnitude trimmed, which in turns bounds the influence of these outliers that are the most likely samples to distort θ_j. This step requires having the data samples of the mini-batch in advance, since it trims the covariance matrix XX^T. Seeing that the model itself could still be corrupted by an excess of outliers, assigning an equal weight to such a bad model could yield a bad model, therefore another filtering technique is applied after the learner observes a whole mini-batch. In it, an approximation of the geometric median is calculated iteratively after each batch using Stochastic Gradient Descent and an update rule with adaptive weights which is given by:

$$\hat{\theta}_{t+1} = (1 - w_t)\hat{\theta}_t + w_t\theta_{t+1}$$

where $w_t = 2\eta_t/||\hat{\theta}_t - \theta_{t+1}||$ and η_t is a predefined step size.

Extending the algorithm to an online distributed learning is straightforward; rather than sequentially streaming batches of data, each learner in a distributed system is assigned a mini-batch in parallel. Our approach assumes a distributed online learning setting where the samples are streamed one by one. In such a setting, there is not requirement to have the complete batch of samples present at the start of a synchronization phase. We therefore apply a single aggregation operation which also yields robust parameter estimation. We compare our method with ROL later in the experiments section.

3 Method and Theoretical Analysis

3.1 Overview

We consider the problem of online machine learning as an instance of a randomized convex optimization problem, that is, we focus on solving problems of the form

$$w_1^*, \ldots, w_T^* = \underset{w_1, \ldots, w_T \in W^T}{\operatorname{argmin}} \sum_{t=1}^{T} \ell(w_t, x_t, y_t),$$

using linear models where the domain $W \subseteq \mathbb{R}^d$ is a convex set that represents the hypothesis space, $\ell : W \times X \times Y \to \mathbb{R}_+$ is a convex loss function and $(x_t, y_t) \sim D$. In a linear model, it is assumed that the relationship between the input $x \in \mathbb{R}^d$ and the output $y \in \mathbb{R}$ is linear, therefore the prediction $\hat{y}$ is computed from a linear combination of the input $x \in \mathbb{R}^d$, i.e. $\hat{y} = \sum_{i=0}^{d} w_i x_i$. In a machine learning context, the solution produced by the convex optimization problem represents the hypothesis that the algorithm has learned by minimizing the loss suffered from the prediction of the observed samples.

The data samples arrive in multiple streams from a distribution $D : X \times Y \to [0, 1]$ where $X \subseteq \mathbb{R}^d$ comprises the samples feature space and $Y \subseteq \mathbb{R}$ is the output space. The range of Y varies depending on the target learning task, for instance, for regression $Y = \mathbb{R}$ whereas for classification $Y = \{-1, +1\}$.

Is is assumed that the nodes responsible for generating the data streams fail occasionally with a given failure probability $P_{failure}$ and recover with a recovery probability $P_{recoveary}$ too. During their failure periods, any feature x_i of an incoming sample x is perturbed by noise with probability P_{noise}. The data samples are perturbed independently using additive Gaussian noise by an adversary such that $x_i = x_i + \xi$ where $\xi \sim N(0, \sigma)$ for some σ whose value determines the magnitude of the noise.

In general, in an online setting the distribution D may vary over, but this scenario is not addressed separately nor does the analysis apply if concept drifts occur in the distribution. However, adapting to concept drifts is an inherent behavior in the proposed approach.

To handle multiple distributed streams, we assume that k distributed learning

nodes $n_1, ..., n_k$ have been put in place, each maintaining its own local model $w_{t,l} \in \mathbb{R}^d$, $t \in [T]$ denotes a discrete point on the time horizon, $l \in [k]$. We assume that all data instances are sampled independently and in parallel from D.

For an incoming instance $(x_{t,l}, y_{t,l}) \sim D$ arriving at node $l \in [k]$ at time t, where y is unknown at the time of arrival, an in-place prediction task of an instance is defined at node l, which produces an output $\hat{y}$ based on the local model $w_{t,l}$ and feature vector x, independently of the other nodes. The true value y of the sample is subsequently revealed to the learner by an Oracle. We also define a performance measure of the collective local models using the same convex loss function ℓ defined above. Such a loss function outputs positive values as penalties for each prediction and the goal of the learning framework is to fit a hypothesis $W \in \mathbb{R}^d$ that minimizes the cumulative loss values incurred across all the learners. Expressed differently using the concept of regret, the objective is to minimize the value of the regret whose definition is as follows:

Definition. ***Regret*** *at time T is the difference between the cumulative loss incurred up to T by a given learning algorithm and that of the best learning algorithm in hindsight:*

$$R_T = \sum_{l=1}^{k} \sum_{t=1}^{T} l(w_{t,l}, x) - l(w^*, x),$$

where

$$w^* = \underset{w \in W}{\operatorname{argmin}} E_x(\ell(w, x))$$

3.2 Theoretical Background

Before discussing the approach in depth, let us first give some background information about the influence of a distributed learning setting on the regret value.
Let us show that in terms of the learning quality, if scalability or decentralization challenges are not taken into account, a single learner achieves the best regret bound. Suppose, for the sake of argument, that there exists one hypothetical supercomputer that is k time as fast as a single learner, if a serial version of a gradient-based learning algorithm is executed on this node, a regret value that is bounded by $O(\sqrt{m})$ is attained (e.g., [5]). Intuitively, this bound is optimal since it retains information on all the incoming data while still being k times faster, as opposed to k slower learners each observing only a fraction of the data. On the other extreme, the worst regret bound possible is reached when each node $i \in [k]$ runs a serial and isolated instance of the same learning algorithm in parallel and does not exchange any information with the other nodes. In this case, if we assume

that a single learner received m/k of the instances, the regret bound for a single node is given by $\sqrt{m/k}$, and for all k nodes the total regret is then $\sqrt{mk}$, worse by a factor of $\sqrt{m}$ than the ideal setting under the premise that no concept drift occurs.
This worst bound that is reached when no communication exchange takes places implies that the need for communication among the nodes in a distributed parallel setting is essential for improving the regret bound.

Seeing that the amount of communication between the nodes is constrained by the network bandwidth, it is imperative to keep the communication amount as small as possible, for instance by communicating a hypothesis that is linear in the data dimensionality d. The network delay, as well as the complexity of the aggregation approach, are crucial factors for attaining a good updated model. In the proposed synchronization scheme, the information that the learners transmit over the network are their local models which are represented as vectors of length d over the Eulidean space, d denotes the data dimension.

Now let us give a description of the exact synchronization scheme proposed in this thesis for an online distributed setting. A synchronization round is initiated by each learner after it has observed b samples, where b is an input parameter that indicates the batch size. The learners do not directly exchange information with each other, they rather communicate their models to a designated medium which we called the coordinator. Each learner transmits its own local model $w_{t\cdot b,l}$ to the coordinator, $l \in [k]$ denotes the learner index and $t \in \mathbb{N}$ the index of the synchronization round. The coordinator in turn produces a new local model w^{σ} based on all models $w_{t\cdot b,l}$ and sends it back to be substituted as each learner's model from that point until the next synchronization phase. A synchronization scheme within which a periodic exchange of information occurs after a fixed number of samples is known as static synchronization. The operator σ whose performance will be studied in this thesis is given by the Radon point of the local hypotheses.

A similar synchronization scheme has been described in Dekel et al. [9] with a variation in the aggregation scheme. In particular, in their paper, the models are aggregated using their arithmetic mean, which is also called the average point, an aggregation operator that helped establish a regret bound that is not far off from the ideal scheme. We refer to this point as the average point as well. This point achieves an asymptotically optimal regret bound of the ideal regret bound $\sqrt{(m)}$.

The proposed algorithm in this thesis employs the same mini-batching technique from Dekel et al. [9]. The term mini-batching is motivated by the mini-batch gradient descent algorithm, which falls somewhere between Batch Gradient Descent and Stochastic Gradient Descent in terms of the number of steps required in order to perform an update step in the gradient. For instance, gradient descent

goes through all the samples in a given dataset before it performs an update step, whereas stochastic gradient descent performs an update after each sample observation. In contrast, mini-batching performs an update step every b samples. In the proposed algorithm herein, one sample is required to perform an update step on a local gradient, but in order to update the global gradient the number of required samples amounts to $b \cdot k$ from all k learners.

Now that the exact synchronization scheme has been described, the problem in fig. 3.1 can be explained. It illustrates a problem that can arise as a result of a slow network and/or a slow aggregation procedure.

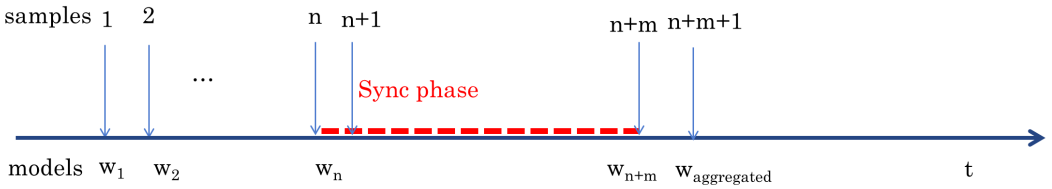


Figure 3.1: n denotes the number of samples required to start the synchronization phase, m the number of discarded samples. s_i denotes the data samples and w_i denotes the updated model after observing sample s_i

Fig. 3.1 describes the problem that may arise in any given synchronization scheme, specifically, in the proposed approach it occurs during the synchronization phase which is the time from which a learner initiates synchronization with the coordinator and until the time it receives the global model. During this phase, data samples continue to arrive, but when the aggregated model arrives at the local node, the updates of these samples are overwritten. In the figure, this corresponds to the updates incurred by the samples $s_n, \ldots, s_{n+m}$. At this point, the synchronized local model is an aggregation of all local models up until samples s_n only and data samples between n and $(n + m)$ don't influence the aggregated model at time $(n + m + 1)$ and are therefore unaccounted for. Depending on the arrival rate of the samples, the complexity of the aggregation and the network delay, m could possibly be a large value. This rather has an adverse effect on the quality of the hypothesis. It is therefore imperative to keep the complexity of the synchronization procedure as low as possible.

3.3 Radon Point

Since the Radon point lies in the core of this approach, first the essential theoretical background related to this point is presented to the reader.
In the field of geometry, the Radon point is defined for a set of points from a convexity space. Here, it is always assumed that the set is over $\mathbb{R}^d$.

To define a Radon point of a set of points $S \subset \mathbb{R}^d$ in the Euclidean vector space, let us first define Radon Partitioning.

Definition. *A **Radon partition** of a set S is two disjoint sets A and B such that the intersection of their convex hull is not empty, i.e., $A, B \subset S$ such that $A \cup B = S$ and $A \cap B = \emptyset$ but $\langle A \rangle \cap \langle B \rangle \neq \emptyset$ ($\langle A \rangle$ denotes the convex hull of the set A).*

Given such a partitioning, what follows is a definition of the Radon point for which it exists.

Definition. *For this set S, a **Radon point** is any point $\tau \in \mathbb{R}^d$ such that there is a Radon partitioning A, B of S and $\tau \in \langle A \rangle \cap \langle B \rangle$. In other words, assuming that a Radon partition exists for a set S, a Radon point is any point in the intersection of the convex hulls.*

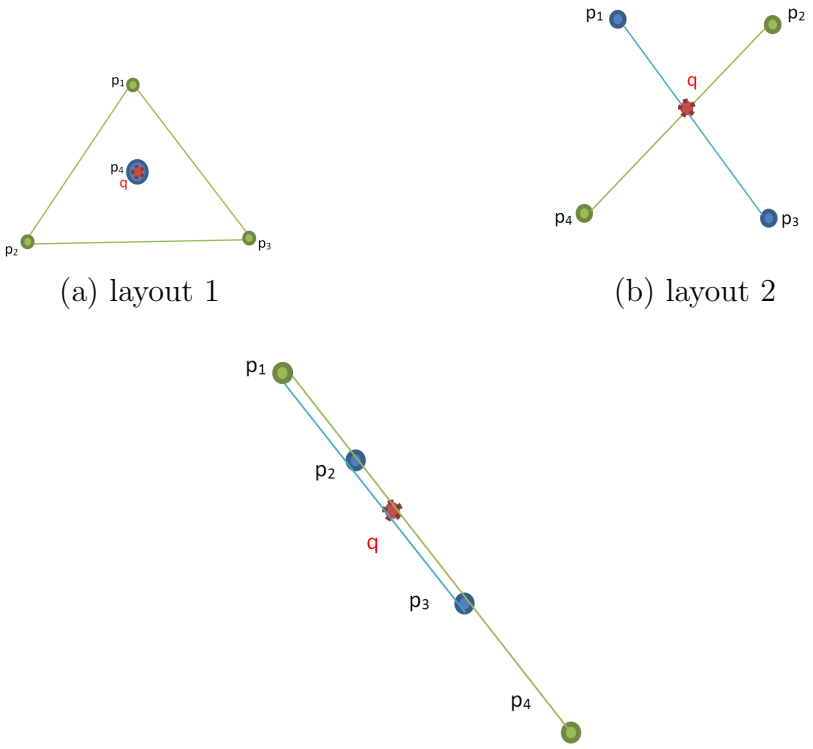
The following theorem states the condition for the existence of a Radon partitioning and therefore a Radon point.

Theorem 2 (Radon [23])**.** *Let $S \in \mathbb{R}^d$ be a set in the d-dimensional Euclidean space. If a subset $X \subseteq S$ contains at least $d + 2$ elements, then there exists a Radon partitioning for the set X, i.e.,*

$$A \cap B = \emptyset, \langle A \rangle \cap \langle B \rangle \neq \emptyset$$

Radon's theorem states that given any set $S \subseteq \mathbb{R}^d$ of $d + 2$ points, S can be partitioned into two disjoint sets such that their convex hulls intersect, therefore the existence of a Radon point is guaranteed for such a set.

Let us give intuition for the theorem by visualizing the case in the two-dimensional space. When the dimension is two, any set of four points in the plane, depending on the points layout, can be partitioned in one of three ways: (a) either one point lies on the convex hull of the other three points, or (b) the four points form the vertices of a convex quadrilateral, in which case the diagonals are the intersecting convex hulls, or (c) the four points are collinear, i.e., lie on the same line. The partitioning then alternates between the four points and the intersection of the convex hulls is the points on the line between p_2 and p_3.



(a) layout 1 (b) layout 2

(c) layout 3

Figure 3.2: The three layouts of a four-point set and their corresponding Radon points in the two-dimensional space. The point q in the intersection of the hulls denotes the Radon point. In the first layout, this is exactly the point itself of the singleton partition

The proof of Radon's theorem also provides a method for constructing a Radon point. Since such a point is a main ingredient in the proposed algorithm, its proof is presented briefly herein.
Let $S = \{x_1, \ldots, x_{d+2}\}$ be any set of $d+2$ points in $\mathbb{R}^d$, it needs to be shown that there exists $\alpha_1, \ldots, \alpha_{d+2}$ coefficients such that not all of them are zero and that satisfy:

$$\sum_{i=1}^{d+2} \alpha_i x_i = 0, \sum_{i=1}^{d+2} \alpha_i = 0$$

Consider the $d+1$ points $x_2 - x_1, x_3 - x_1, \ldots, x_{d+2} - x_1$. These points are linearly dependent since any $d+1$ points in a d-dimensional space are linearly dependent.

Therefore, there exists $\beta_3, \ldots, \beta_{d+2}$ such that:

$$x_2 - x_1 = \sum_{i=3}^{d+2} \beta_i \cdot (x_i - x_1).$$

Rearranging the terms yields

$$(\beta_3 \cdots + \beta_{d+2} - 1)x_1 + x_2 + (-\beta_3)x_3 + \cdots + (-\beta_{d+2})x_{d+2} = 0$$

Setting $\alpha_1 = \beta_3 + \cdots + \beta_{d+2} - 1, \alpha_2 = 1, \alpha_i = -\beta_i$ for $i \geq 3$, then the coefficients have been found .
To find the partitioning, we set $P_1 = \{i|a_i > 0\}, P_2 = \{i|a_i < 0\}$, and then we set the sets that correspond to this partitioning to $A_1 = \{x_i|i \in P_1\}$ and $A_2 = \{x_i|i \in P_2\}$. That is, one set of the partitioning contains the elements whose coefficients are negative and the other contains the elements whose coefficients are positive.
Set $S = \sum_{i \in P_1} \alpha_i$, it also follows that $S = -\sum_{i \in P_2} \alpha_i$, the point $x = \sum_{i \in P_1} \frac{\alpha_i}{S} \cdot p_i$ lies in the convex hull of A_1, and since

$$\sum_{i \in P_1} \alpha_i p_i + \sum_{i \in P_2} \alpha_i p_i = 0,$$

it holds that $x = \sum_{i \in P_2} -\frac{\alpha_i}{S} p_i$, therefore x lies in the convex hull of A_2 as well.

Let us now proceed to describe the construction of a Radon point.
Given a set S, $|S| \geq d + 2$, finding a Radon Point of this set requires constructing a matrix A whose elements are the models' coefficients and then solving the set of formulas given by:

$$\underset{((d+2)\times k)}{A} \times \underset{(k\times 1)}{x} = \underset{((d+2)\times 1)}{b}$$

where $b = [0, \ldots, 0, 1]^T$ and k is the number of learners.
A solution is guaranteed to exist based on Radon's theorem whose proof has been presented earlier.
Let us give concrete example that matches the setting assumed. let $d = 3$, since at least $d + 2$ models are needed in order for a Radon point to exist, we also assume 5 nodes and their corresponding models $\{w_1\}_{i=1}^5$ where $w_i, \in R^3$, i.e., $d = 3, r = d + 2 = 5, k = 5$. Let
$w_1 = \begin{bmatrix} 1 & 0.2 & 0.4 \end{bmatrix}$, $w_2 \begin{bmatrix} 0.1 & 0.3 & 0.35 \end{bmatrix}$, $w_3 \begin{bmatrix} 0.2 & 0.15 & 0.4 \end{bmatrix}$, $w_4 \begin{bmatrix} 0.17 & 0.15 & 0.23 \end{bmatrix}$, $w_5 \begin{bmatrix} 0.11 & 0.19 & 0.24 \end{bmatrix}$

then the weight vectors w_i make up the columns of the matrix A, and two

additional rows are added to ensure that a solution exists:

$$A = \begin{bmatrix} 1 & 0.1 & 0.2 & 0.17 & 0.11 \\ 0.2 & 0.3 & 0.15 & 0.15 & 0.19 \\ 0.4 & 0.35 & 0.4 & 0.23 & 0.24 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

where the last row is added to guarantee a non-zero ($\vec{0}$) solution. The solution vector $b = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ and then we solve for x: $Ax = b$.
The complexity of calculating a Radon point is $O(r^3) = O(d^3)$ since it requires solving a set of $d + 2$ linear equations as described above.

It has already been briefly mentioned that the Radon point is the synchronization method employed in this algorithm. The above case finds the Radon number when the number of points is exactly $d + 2$, but it does not solve the case when there are more points (i.e., learners) than the Radon number. However, to find a Radon point over all the points rather than a subset of size $d + 2$, we use the algorithm described in the paper Clarkson et al. [7]. In this paper, the authors have devised a Monte-Carlo algorithm called ***Iterated Radon Points*** for approximating center points. Their idea is motivated by the fact that a Radon point of a set of $d + 2$ is a $\frac{2}{d+2}$ center of that set. This is due to a property of a Radon point, which states that given a Radon partitioning A, B of a set P, then any half-space containing a Radon point q contains at least one point from A and one point from B. This means that the other side of the hyperplane has at most d points, making the splitting ratio of the set by its Radon point at most $\frac{d}{d+2}d$.

Definition. *A point $c \in \Re^d$ is a $\beta -$ center point of P if every closed half-space containing c has $\beta \cdot n$ points of the set.*

Below, the Iterated Radon Points algorithm is presented and then shown how it is applied in the context of the proposed online learning scheme.

Put into words, given a set $P \in \mathbb{R}^d$, the Iterated Radon Points algorithm first constructs a complete $d+2$-ary tree, i.e., each internal node is connected by edges to $d+2$ children. The point assignment to these nodes is determined iteratively in a bottom-up fashion. In the first step, the set of points P are assigned randomly to the leaf nodes. At the second level of the tree, the algorithm computes the Radon point for each child subset of size $d+2$ and assigns it randomly to a parent node adding a new level in the tree. By Radon's theorem, the selection of the size $d+2$ for the subsets guarantees the existence of a Radon point. These intermediate Radon points are divided again into subsets of $d+2$ points. The method is applied

Algorithm 1: Iterated Radon Points algorithm

Input: A set $P \subset \mathbb{R}^d$ of size L
Output: A Radon point of P

1. Construct a complete balanced (d+2)-way tree T of L leaves (assume that L is a power of $(d+2)$)
2. For each leaf l of T, choose a point from P uniformly at random and independently of the other leaves and assign it to l
3. Evaluate tree T in a bottom up fashion as follows:
4. For each internal node $v = p_i^j$ at level $1 \leq j \leq \log_{d+2}|P|$, $0 \leq i \leq \frac{|S|}{(d+2)^j}$ compute and assign to v the Radon point of its $d+2$ child nodes
5. Output the point that has been assigned to the root of T

recursively to each level of the tree until it finally reaches the root node. The height of such a tree is $\log_{d+2}(|P|)$ and the number of nodes at level i is $\frac{|P|}{(d+2)^i}$. The Radon point assigned to the root node is the ultimate Radon point that the algorithm outputs. From our perspective, this is an Iterated Radon Point and is exactly Radon point when the height of the tree is one. We will simply refer to it as Radon point henceforth.

Let us have a look at a concrete example in the three-dimensional space. In particular, let $d = 3$, the Radon number $r = d + 2 = 5$ and the set size $|P| = (d + 2)^3 = 125$. The tree height is therefore $h = \log_r(|P|) = 3$.

The complexity of finding a combined model using the proposed approach is equal to the complexity of the Iterated Radon Points algorithm whose complexity is given by $O(d^2 \cdot L)$, L the size of the initial set. In this scheme, L is the number of local models and corresponds to the number of learner nodes k. The complexity can be established as follows:
calculating Radon point for a set of size $r = d + 2$ requires solving a set of $d + 2$ linear equations as described in the example illustrated in fig. 3.3. The complexity of finding such a solution is $O(r^3)$, and the number of solutions calculated is the number of the internal nodes in the tree. The total complexity is therefore :

$$T(L,d) = r^3 \sum_{i=1}^{\log_r(L)} \frac{r^i}{r} = r^3 \sum_{i=1}^{\log_r(L)} r^{i-1} = r^3 \frac{r^{\log_r(L)}}{r-1} = r^3 \frac{L}{r-1} = O(d^2 \cdot L))$$

Note that the algorithm complexity does not depend on the dataset size, which could be infinite in an online streaming scenario. On the other hand, to compute the average point of L points it takes $O(d \cdot L)$.

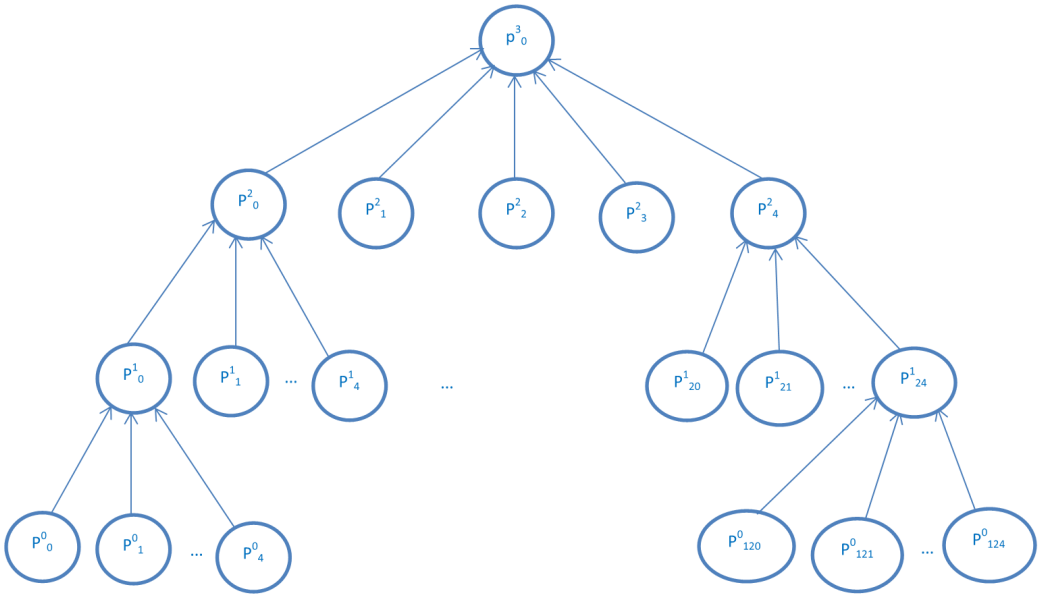


Figure 3.3: The leaves $\{p_i^0\}_{i=1}^n$ comprise the initial set of points in $\mathbb{R}^3$, p_i^j indicates the i^{th} Radon Point at level $j > 1$ in the tree
Iterated Radon Points example

3.3.1 PAC learnability

One of the objectives of this research is to establish probabilistic error guarantees in an online distributed scenario with intermediate synchronization phases. Indeed, it was shown that the algorithm achieves a Probably Approximately Correct (PAC Valiant [25]) type of bound, with an exponentially decreasing error probability as more synchronization phases are completed. A decreasing error probability is a natural consequence as the learners observe more samples, as well as the learners synchronize periodically. In fact, synchronization is an operation that may be regarded as the process of observing more samples as well. Below is a definition of the PAC framework.

Definition. ***PAC learnability*** *A concept class C is PAC-learnable by a hypothesis class H if and only if for any $c \in C$, $1 > \epsilon, \delta > 0$ and any distribution D over a sample space X, there exists a polynomial time learning algorithm A that receives as input ϵ, δ and m labeled data samples, drawn independently at random from D, and outputs a hypothesis $h \in H$ such that*

$$\mathbb{P}[|h(x) - f(x)| > \epsilon] < \delta,$$

when m is polynomially bounded by $1/\delta$ and $1/\epsilon$.

The above formula can be interpreted as follows: if the algorithm A is given m many samples, then with probability at least $1-\delta$, A returns a hypothesis $h \in H$ whose error $Error(h) \leq \epsilon$.

PAC learning is a powerful paradigm in the field of machine learning. Its strength stems from its capability of inferring knowledge on the generalization error over a probability distribution D, i.e., the error probability on the unseen data that are drawn from D, using the sample error only. This inference for unseen samples provides a reliability measure on the learning algorithm that is attained from seeing a certain number of samples. Note that PAC does not make any assumption on the distribution D. PAC learnability also relaxes the requirement on the correctness of the learning algorithm using the error probability ϵ. However, the smaller the value of δ is, the stronger the guarantee obtained on the quality of the learning algorithm. δ denotes the confidence parameter where the confidence is given by $1-\delta$.

Classical PAC learnability definition pertains to offline learning in which the complete dataset is available when the learning starts, as can be deduced from the definition above. In this work, the classical PAC framework was extended such that it enabled us to draw similar guarantees to the online scenario.

3.4 Method Description

Now that the necessary background related to the method has been presented, the algorithm implementing it is now given, which is called **Online Radon Machine**.

Algorithm 2: Online Radon Machine algorithm

Learner l**:** before synchronization phase $t \in \mathbb{N}$
at observing sample x_i:
  predict a label $\hat{y_i} = f(x_i, w_{t+i})$
  at observing y_i:
    update $w_{t+i+1,l} = \varphi(w_{t+i,l}, x_i, y_i)$
    if $i \mod b = 0$ **then**
      send $w_{t+i,l}$ to Coordinator
Coordinator at synchronization phase t:
  Receive local models $P = \{w_{t,l}\}_{l=1}^{k}$
  $w =$ **Iterated_Radon _Points**(P)
  send w to all the learners $[k]$ and set $w_{t,1} \ldots, w_{t,k} = w$

The pseudo-code in Algorithm 2 describes the algorithm on both the learner and coordinator nodes. When a learner $l \in [k]$ observes a sample x_i before synchronization phase t, it calculates a prediction score using the dot product $\langle x_i, w_{t+i,l} \rangle$. Note that w_t is the aggregated model received from the coordinator at synchronization phase t and w_{t+i} is the updated local model after observing the i^{th} sample in the new batch. Based on this score, the learner decides the label $\hat{y}_i$. For instance, for linear regression, $\hat{y}_i$ is equal to the prediction score itself and for binary classification it decides $True$ if the prediction score is positive and $False$ otherwise. When $t = 0$ which is the time before any synchronization occurred yet, w_l is simply the local hypotheses learned from the samples routed at node l

When the true label y_i is subsequently revealed to the learner, the learner updates its model $w_{t+i,l} = \varphi(w_{t+i-1,l}, x_i, y_i)$, where the update operator is determined by the update rule and is defined by $\varphi : \mathbb{R}^d \times X \times Y \to \mathbb{R}^d$. Examples of such an update rule include Stochastic Gradient Descent, which is the rule employed in the experiments as well. Note, however, that according to the black-box property that the algorithm satisfies, no restriction is set on what update rule is used given that they meet the conditions described earlier, implying that other such update rules may be used as well without causing any alteration in the overall algorithm.

When the learner has observed b samples, it initiates a synchronization procedure by sending its model $w_{t+b,l}$ to the coordinator. Once the coordinator received the models of all the learners, it invokes Iterated Radon Points algorithm to compute their Radon point and then sends back the result to each of the learners. The operator induced by the synchronization protocol together with Iterated Radon Points is defined as $\sigma : \mathbb{R}^{d \times k} \to \mathbb{R}^{d \times k}$.

Utilizing Radon Point as a method for model synchronization has so far only been researched in a batch mode parallel setting in Kamp et al. [20], for which the authors have established Probably Approximately Correct (PAC) guarantees. Note that their analysis applies to uncontaminated data, as does our analysis and for now the analysis of a contaminated data stream is left for future work.

It has already been mentioned in the introduction that the proposed method is based on the algorithm in Kamp et al. [20], now let us elaborate on their algorithm since it constitutes the basis for the algorithm in this thesis as well. Even though the algorithm there is suited for batch scenarios, it can be extended to online scenarios thereby matching the setting being tackled in this thesis. In the batch scenario, Let A be a serial learning algorithm, A is instantiated separately on c distributed nodes each of which is handling a separate subsets of the data whose samples arrived from the same distribution. As a result, c models that were learned independently are obtained at the end of the learning process. In order to increase the confidence of a locally learned hypothesis, the learners send

their local hypotheses to the coordinator, where it invokes the Iterated Radon Points algorithm whose input are these c hypotheses. The algorithm combines these weak independent hypotheses by iteratively replacing them with their Radon point thereby creating a stronger global hypothesis. If the initial given confidence parameter to the algorithm is δ for each learner after seeing b samples as yielded in a classical PAC algorithm, then they achieved an improved confidence of

$$(r\delta)^{2^h} \tag{3.1}$$

as proved by Kamp et al. [20].

The online algorithm utilizes the same aggregation algorithm in the same manner, but with no assumption on the length of the data streams, it rather treats them as streams of an indefinite length and therefore the base batch algorithm can be applied infinitely many times where each time the confidence keeps improving.

The difficulty in conceiving probabilistic bounds in the online mode, which are analogous to the batch mode, lies in obtaining subsequent improved bounds following the first one. In particular, usually the probabilistic bound is obtained as a function of the number of instances rather than an aggregated model. To solve this problem, we devise a method in which the improvement from synchronization is simulated by an actual number of instances, i.e., given the improvement, we inversely find the number of instances required for it, thereby simplifying our task and treating the improvement as if it were equivalently obtained with a larger number of samples than the actual number. In the online setting, we still have to examine how these concepts for a fixed-size data can be modified so that PAC learning may be achieved.

3.5 Theoretical Analysis

3.5.1 Theoretical Background

Gärtner and Missura [14] have shown that a generalization of the Iterated Radon Points algorithm can be used to approximate the maximum of a quasiconcave function in a convexity space with high probability under certain assumptions on the sampling distribution. The following property which holds with regard to Radon Point in a convexity space:
For any Radon point q of a set S and any convex function f, there exists at least two points $a, b \in S$ such that $f(q) < f(a)$ and $f(q) < f(b)$. Using this property of a Radon point, the following inequality can be proved by applying the union

bound:

$$P[f(q) \leq \theta] \leq (rP[f(X) \leq \theta])^2 \tag{3.2}$$

where $\theta \in \Re$ and X is a sample from a fixed distribution over the convexity space and Y is the Radon point of a set of size r drawn $i.i.d$ from the same distribution.

3.5.2 PAC for the Radon point

By iteratively applying formula 3.2 at each height in the Iterated Radon Points tree, Kamp et al. [20] have established the following theorem:

Theorem 3 (Kamp et al. [20])**.** *Given a probability measure P over a hypothesis space H with a finite Radon number r, let X represent the random variable that represents any point drawn according to P. Let τ_h denote the random variable that represents the output of the parallel Radon machine after h iterations with base hypotheses drawn according to P. Then for any convex function $Q : H \to \mathbb{R}$ and all $\epsilon \in \mathbb{R}$ it holds that*

$$\mathbb{P}[Q(\tau_h) \geq \epsilon] \leq (r\mathbb{P}[Q(X) \geq \epsilon])^{2^h} \tag{3.3}$$

They also have established a number of samples $N_0 \in \mathbb{N}$ for which the PAC guarantees on the confidence hold. What follows is a description of their method. Assume that there exists an algorithm A that finds an approximate solution for the optimization problem in the batch mode, namely

$$w^* = \underset{w \in W}{\operatorname{argmin}}\, \ell(w).$$

A takes as input a parameter $N \in \mathbb{N}$ and outputs a solution w that is distributed according to a distribution $D_N : W \to [0, 1]$. Such a distribution D_N satisfies that for all $\epsilon > 0$ and $\Delta \in (0, 1)$, there exists N_0 such that $\forall N \geq N_0$, it holds that

$$P_{w \sim D}(|f(w) - f(w^*)| > \epsilon) < \Delta$$

An algorithm that provides such a probabilistic guarantee is called consistent and in particular the algorithm A is consistent. Additionally, we are interested in algorithms whose runtime complexity is at most polynomial in the size of the sample set N, i.e., $T(A) \in \mathcal{O}(N)^{c_1}$ for some constant $c_1 \in \mathbb{N}$. A, in itself, is a serial algorithm and yields these guarantees for a single learner after observing N samples. In attempting to decrease the runtime of the algorithm, c parallel processors are employed in parallel, on each of which runs an instance of the algorithm A on a fraction N/c of the data. Each instance then learns a solution w_i

in parallel and independently of the other learners. This parallelization degrades the guarantees of δ, ϵ that were obtained for the parameter N, as already has been showed in the discussion regarding a distributed setting. Intuitively, it is true because the more data samples a learner sees, the better it can learn the underlying model. In order to mitigate the parallelization influence, the solutions of the learners are combined using the Iterated Radon Points algorithm.

For a Radon point $\tau(W)$, it holds that for all measures P over $W \in R^d$, a quasi-convex function f, all $w \in W$ and all $\epsilon \in \mathbb{R}_+$ that

$$P(|(f(\tau) - f(w^*)| > \epsilon) < (rP(|f(w) - f(w^*)| > \epsilon))^2.$$

Running the Iterated Radon Points algorithm on the set of solutions obtained by the learners improves this bound exponentially by the height of the tree, namely, the following enhanced bound is obtained:

$$\mathbb{P}[(|(f(\tau) - f(w^*)| > \epsilon)] < (r\mathbb{P}[|f(w) - f(w^*)| > \epsilon])^{2^h}, \tag{3.4}$$

where $h = \log_r |W|$ as proved by Kamp et al. [20].

For the analysis, let us set ϵ as a fixed value and then examine the dependency of Δ on the value of N. (Note that the improvement in ϵ is similar when the value of Δ is fixed). Combining the set solutions W produced by A, each satisfying a confidence parameter δ, yields an enhanced confidence parameter given by $(r\delta)^{2^h}$. In order to ensure that the value of the confidence parameter has been improved, we require that $(r\delta)^{2^h} < \delta$, which restricts the range of δ to:

$$\delta < r^{-\frac{2^h}{2^h - 1}} \quad . \tag{3.5}$$

Due to the consistency property that the algorithm A satisfies, the existence of a parameter $n_0 \in N$ such that $\forall\ n \geq n_0$ the confidence holds is guaranteed, and accordingly, the parameters N and c have to be set such that $N/c \geq n_0$. The above analysis proves that the parallelization scheme described above is also consistent, i.e., it maintains and even improves the individual probabilistic guarantees that were obtained separately by each learner.

As stated earlier, the goal is to devise a method that would convert the guarantee model to an online PAC model. To the best of our knowledge, there exists no direct conversion from an offline PAC guarantee to an online PAC guarantee. PAC, by its definition, requires a predefined number of samples for an error and confidence guarantees to be established. It is possible, however, to transform an online mistake bound to a PAC guarantee due for instance to the power of the

online mistake bound model.

3.5.3 Converting Regret Bound to PAC

In what follows an online PAC guarantee that relies on conversion from the Regret bound is established. Let us first introduce some definitions and notations that will be used in the conversion. The setting of the learning problem in general is defined by an online learning algorithm that receives at each discrete point in time $t \in \mathbb{N}$ an example $z_t \in \mathcal{Z}$ from an input distribution $\mathcal{D}$ and provides an output based on its local model $f_t \in \mathcal{H}$. The quality of the output is measured by a convex loss function $\ell \colon \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$.

Another key ingredient that has been used in the proof is Hoeffding's inequality [17], which in general is an important inequality in learning theory. The inequality provides a bound on the probability of a sum of independent variables deviating from the expected value of their sum. More formally, given n independent random variables $Z_1, \ldots, Z_n$ where $Z_i \in [a, b], i \in [n]$, let $\bar{Z} = \frac{1}{n}(Z_1 + \cdots + Z_n)$ be the empirical mean of the random variables, then for all $t > 0$

$$\mathbb{P}\big[\bar{Z} - E[\bar{Z}] \geq t\big] \leq \exp(-\frac{2n^2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2})$$

When the variables Z_i are bounded in the interval $[0, 1]$, then namely it holds:

$$\mathbb{P}\big[\bar{Z} - E[\bar{Z}] \geq t\big] \leq e^{-2nt^2}.$$

For our purpose, if the random variables do not lie on the interval $[0, 1]$, a normalization procedure can be initiated that scales them to this interval and then the inequality in the above form can be used.

For the conversion, we also make use of the Regret value which is easily quantified both offline and online.
Let $X_1, X_2, \ldots$ be random variable representing the loss values of the model with $X_t = \ell(f_t, z_t)$ for $z_t \sim \mathcal{D}$,. Furthermore, for a time horizon $T \in \mathbb{N}$, let

$$\overline{X} = \frac{1}{T}\sum_{t=1}^{T} X_t \ .$$

denote their average and $\mathbb{E}\left[\overline{X}\right]$ the expected value of this average.

Let $R(T)$ be the regret bound of the online learning algorithm, i.e.,

$$\sum_{t=1}^{T} \ell(f_t, z_t) - \ell(f^*, z_t) \leq R(T)$$

for all input sequences $z_1, \ldots, z_T \in \mathcal{Z}$ and all models $f^* \in \mathcal{H}$ (including the optimal model in hindsight).

Using Hoeffding's inequality, we obtain

$$\begin{aligned}
&\mathbb{P}\left[\overline{X} - \mathbb{E}\left[\overline{X}\right] \geq \theta\right] \leq e^{-2T\theta^2} \\
&\Leftrightarrow \mathbb{P}\left[\overline{X} \geq \theta + \mathbb{E}\left[\overline{X}\right]\right] \leq e^{-2T\theta^2} \\
&\Leftrightarrow \mathbb{P}\left[\overline{X} \geq \theta + \mathbb{E}[\frac{1}{T}\sum_{t=1}^{T} \ell(f_t, z_t)]\right] \underbrace{\leq}_{\sum_{t=1}^{T} \ell(f_t,z_t) \leq R(T) + \sum_{t=1}^{T} \ell(f^*,z_t)} e^{-2T\theta^2} \\
&\Leftrightarrow \mathbb{P}\left[\overline{X} \geq \theta + \frac{1}{T}(R(T) + \sum_{t=1}^{T} \ell(f^*, z_t))\right] \leq e^{-2T\theta^2}
\end{aligned}$$

We make the following two assumptions:

1. There exists a perfect classifier f^*, i.e., $\ell(f^*, z) = 0$ for all $z \in \mathcal{Z}$.

2. The loss of a model decreases over time, i.e., for $t > t'$ it holds that

$$X_t \leq X_{t'} \Leftrightarrow \ell(f_t, z) \leq \ell(f_{t'}, z) \text{ for } z \sim \mathcal{D} \ .$$

Using the first assumption yields:

$$\mathbb{P}\left[\overline{X} \geq \theta + \frac{1}{T}R(T)\right] \leq e^{-2T\theta^2} \ .$$

From the second assumption it follows that:

$$\ell(f_T, z_T) = X_T = \frac{1}{T}\sum_{t=1}^{T} X_T \leq \frac{1}{T}\sum_{t=1}^{T} X_t = \overline{X}$$

and thus

$$\mathbb{P}\left[\ell(f_T, z) \geq \theta + \frac{1}{T}R(T)\right] \leq e^{-2T\theta^2} \ . \tag{3.6}$$

The second assumption indicates that the last model is the best model among all T observed samples, and therefore its loss value is smaller than the average of the loss values. This is a reasonable assumption in a learning algorithm, since ultimately the goal is to improve the models as the learner observes more, even though this still depends on the instance itself. Defining

$$\epsilon_{(T,\theta)} = \theta + \frac{1}{T}R(T) \ , \tag{3.7}$$

and

$$\delta_{(T,\theta)} = e^{-2T\theta^2} \tag{3.8}$$

produces the necessary parameters for defining a PAC guarantee. Note that $\theta \in \mathbb{R}$ can be chosen arbitrarily and thus can realize any desired $\epsilon > 0$ for a given time horizon $T \in \mathbb{N}$.

To see the improvement obtained from only one synchronization, we can apply the bound obtained for δ from the batch mode, which is given by:

$$\delta_1^\sigma = (r\delta_0)^{2^h} \tag{3.9}$$

where δ_0 is the initial confidence parameter given b samples and r is the Radon number. When the first synchronization phase is initiated by the learners and their Radon point is calculated by the coordinator, the confidence in Eq. 3.9 is the improvement obtained. Note that a smaller value of δ is an enhancement in the confidence parameter, since the confidence is given by $1 - \delta$.

Now that each learner has an improved value for the confidence, we would like to find the improved value for δ_i^σ in any subsequent synchronization phase $i > 1$. Since the value of the confidence is a function of the number of instances which does not take into account the improvement obtained from the synchronization, applying the same formula directly again is not possible. It is possible, however, to find a value b' with which the formula for the confidence in 3.8 can be applied. In other words, we emulate the improvement obtained by the synchronization by increasing the number of samples per learners and omitting synchronization. Then in order to apply the formula for the confidence using Eq. 3.9 right before the second synchronization phase, we substitute $b_2 = b' + b$ in Eq. 3.8.

The online Radon algorithm runs k independent instances of an online sequential learning algorithms in parallel in a mini-batching manner. Mini-batching implies that each learner is learning independently online for $b \in \mathbb{N}$ samples before synchronization its model with the other learners. Let $f_{tb}^1, \ldots, f_{tb}^k$ denote the local model before any synchronization phase $t \in \mathbb{N}$ and f_{tb}^σ denote the local model after the same synchronization phase. The aggregated model $f_t^\sigma b$ is then used by all

learners as a new starting point for learning for another b samples. Seeing that this is an online setting, the scheme is repeated indefinitely for any desired duration. We are interested in finding a PAC guarantee for any synchronization phase t or number of samples $T \in \mathbb{N}$ per learner. Assume for simplicity that b divides T, thereby implying that there is an aggregation step in the final round. We are interested in a PAC guarantee for f_T^σ.

Theorem 4. *Let f_T^σ denote a model at time $T \in \mathbb{N}$ that is obtained by aggregating $k \in \mathbb{N}$ models learned in parallel at round $n = \frac{T}{b} \in \mathbb{N}$, b the batch size and b divides T. If the models are aggregated using the parallel Radon machine with a tree height $h = \lfloor \log_r k \rfloor$ for $r = d + 2$, d is the data dimensionality, then the PAC guarantee of f_T^σ is given by the following formula:*

$$\delta_T^\sigma = \prod_{i=1}^{n} (r\delta)^{2^{ih}} = (r\delta)^{\sum_{i=1}^{n} 2^{ih}} \tag{3.10}$$

Proof. We prove the theorem by induction.

Base case $T = b$

The correctness of the base case is established directly by substituting $n = 1$ in Eq. 3.10 and comparing its result to the result obtained in Eq. 3.9.

Induction Hypothesis

Assume that the proposed formula holds when $n = m$, i.e.,

$$\delta_T^\sigma = \prod_{i=1}^{m} (r\delta)^{2^{ih}} = (r\delta)^{\sum_{i=1}^{m} 2^{ih}}$$

Inductive step

Before proceeding to proving the inductive step, note that the improved confidence after synchronization at $T = b$ is equivalent to the confidence of having learned for T_{δ^σ} samples with no synchronization where T_{δ^σ} is given by:

$$T_{\delta_T^\sigma} = -\frac{\ln \delta_T^\sigma}{2\theta^2}$$

This can be established directly from Eq. 3.8 which assumes no synchronization and relies on Hoeffding's inequality:

$$e^{-2T\theta^2} \Rightarrow T = -\frac{\ln \delta}{2\theta^2} \tag{3.11}$$

This equivalent number of samples is what enables us to establish PAC guarantees for any synchronization phase $i \geq 1, i \in \mathbb{N}$ as will be shown shortly.

Now we are ready to show that the proposed formula holds when $n = m + 1$. Using the induction hypothesis, the equivalent number of samples required to obtain the guarantee for $T = mb$ when no synchronization occurs may be found:

$$T_{\delta^{\sigma}_{mb}} = -\frac{\ln \delta^{\sigma}_{T}}{2\theta^2} = -\frac{\ln\left((r\delta)^{\sum_{i=1}^{m} 2^{ih}}\right)}{2\theta^2} = -\frac{\ln(r\delta)}{2\theta^2}\sum_{i=1}^{m} 2^{ih}$$

Before the start of the synchronization round $m + 1$, the learner will have seen b more samples, i.e., the number of samples then is:

$$T_{\delta^{\sigma}_{(m+1)b}} = -\frac{\ln r\delta}{2\theta^2}\sum_{i=1}^{m} 2^{ih} + b.$$

From learning alone, a local learner will obtain the following guarantee at the end of this round (again, by Eq. 3.8):

$$\begin{aligned}
\delta_{(m+1)b} &= e^{-2T_{\delta^{\sigma}_{T}}\theta^2} = \\
&= e^{-2[\frac{-ln(r\delta)\sum_{i=1}^{k} 2^{ih}}{2\theta^2} + b]\theta^2} \\
&= e^{ln(r\delta)\sum_{i=1}^{k} 2^{ih}} e^{-2b\theta^2} \\
&= (r\delta)^{\sum_{i=1}^{k} 2^{ih}} \cdot \delta
\end{aligned}$$

After synchronization, the result from Eq. 3.9 can be naturally applied and thereby we obtain:

$$\delta^{\sigma}_{(m+1)b} = [r\delta(r\delta)^{\sum_{i=1}^{k} 2^{ih}}]^{2^h}$$

What remains to be shown is that this result is equivalent to the result yielded by the given theorem, i.e., we require that

$$\delta^{\sigma}_{(m+1)b} = \prod_{i=1}^{k+1}(r\delta)^{2^{ih}}$$

Below we try to bring the result obtained by induction to a similar form:

$$
\begin{aligned}
&\delta^{\sigma}_{(m+1)b} = [r\delta(r\delta)^{\sum_{i=1}^{m} 2^{ih}}]^{2^h} = \\
&(r\delta)^{2^h}(r\delta)^{2^h \sum_{i=1}^{m} 2^{ih}} = \\
&(r\delta)^{2^h}(r\delta)^{2^h \sum_{i=1}^{m} 2^{(i+1)h}} = \\
&(r\delta)^{2^h}(r\delta)^{2^h \sum_{i'=2}^{m+1} 2^{i'h}} = \\
&(r\delta)^{2^h \sum_{i=1}^{m+1} 2^{ih}} = \\
&\prod_{i=1}^{m+1} (r\delta)^{2^{ih}}
\end{aligned}
$$

as needed. □

Below, this is illustrated for the first and second synchronization phase in order to give some intuition to the reader for the proof presented above. Observe that after a learner has seen b samples we obtain $\delta_0 = e^{-2b\theta^2}$, therefore:

$$
\begin{aligned}
&\delta_0 = e^{-2b\theta^2} \Rightarrow b = -\frac{\ln \delta_0}{2\theta^2} \\
&\delta^{\sigma}_{b} = (r\delta_0)^{2^h} = e^{-2b'\theta^2} \Rightarrow b' = -2^h \frac{\ln(r\delta_0)}{2\theta^2} \\
&\delta_{2b} = e^{-2(b'+b)\theta^2} = e^{-2\left[\frac{-2^h \ln(r\delta_0)}{2\theta^2} + b\right]\theta^2} = e^{2^h \ln(r\delta_0)} e^{-2b\theta^2} = (r\delta_0)^{2^h}\delta_0 \\
&\delta^{\sigma}_{2b} = (r\delta_{2b})^{2^h} = (r(r\delta_0)^{2^h}\delta_0)^{2^h} = \delta^{\sigma}_{b}(r\delta_0)^{2^{h+1}}
\end{aligned}
$$

3.5.4 Interpretation

Let us now interpret the influence of the above bound on the convergence speed. If we examine the number of samples required to reach the same value of the confidence obtained after one synchronization:

$$
\delta^{\sigma}_{b} = (r\delta_0)^{2^h} = e^{-2b'\theta^2} \Rightarrow b' = -2^h \frac{\ln(r\delta_0)}{2\theta^2} = -2^h \frac{\ln \delta + \ln r}{2\theta^2} = 2^h b - \frac{2^h \ln r}{2\theta^2}
$$

To see how significant the improvement is, let us give concrete values to these parameters and plot for a variable value for each one of them.

Recall that the improved batch size is the number of samples that the learner would have needed to observe in order to obtain the same enhanced confidence yielded by the synchronization. In fig. 3.4, there are $k = 625$ parallel learners learning the same hypothesis by observing multiple streams arriving from the same distribution. So while in fact there are k learners and the actual number of

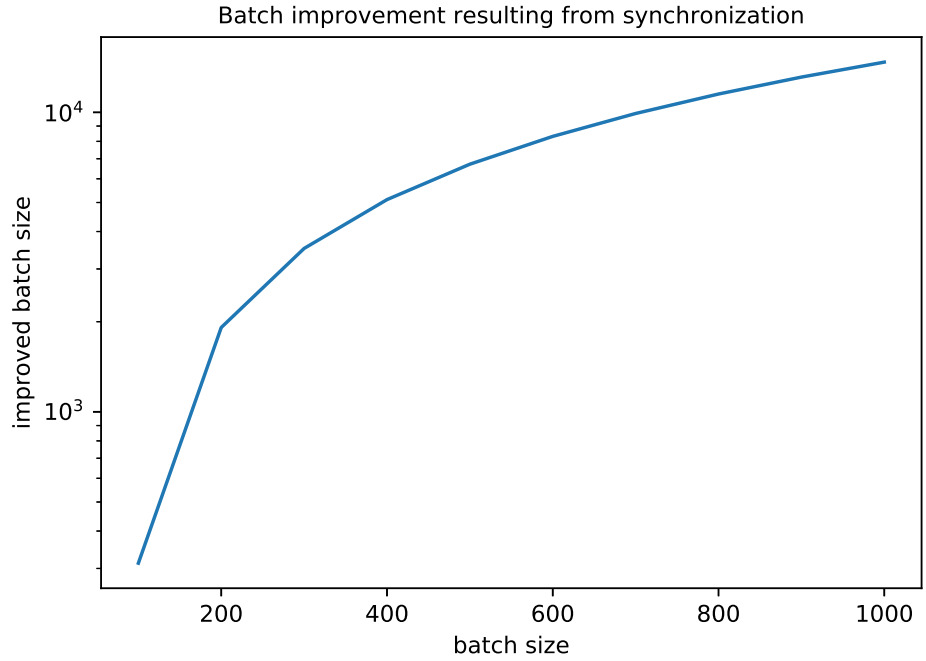


Figure 3.4: The improved batch size for a variable batch size. In this plot, $\theta = 0.1, h = 4, r = 5, k = 5^4 = 625$. The larger the batch size is, the more significant the improvement becomes since this is equivalent to a high number of observed samples per learner and therefore implies a better learned hypothesis. (log scale)

samples observed in the whole system is $k \cdot b$, a central learner alone would have taken a longer time to reach the same hypothesis. In the figures, an iteration refers to a synchronization round. Note that a lower value for the confidence parameter δ indicates better confidence since the confidence is given by $1 - \delta$.

The interpretation of fig. 3.5 and 3.6 is that given these values for a Radon number or a height, the resulting value of the allowed maximum δ as given by eq. 3.5 is the value that would allow for improvement of the confidence parameter in the algorithm. We observe that the higher the value of the height, the higher this value becomes. On the other hand, the higher the Radon number, the lower this maximum value becomes. This can again be attributed to the height; since $h = \log_r(k)$, if the value of r increased then the value of h decreases.

In fig.3.10, the same values from the previous plot are used, but instead of the iteration on the x-axis, the actual number of samples seen by the learner up until that iteration is shown, i.e., $b_0 \cdot t$.

In fig. 3.7, we calculate the confidence parameter δ across different iterations for

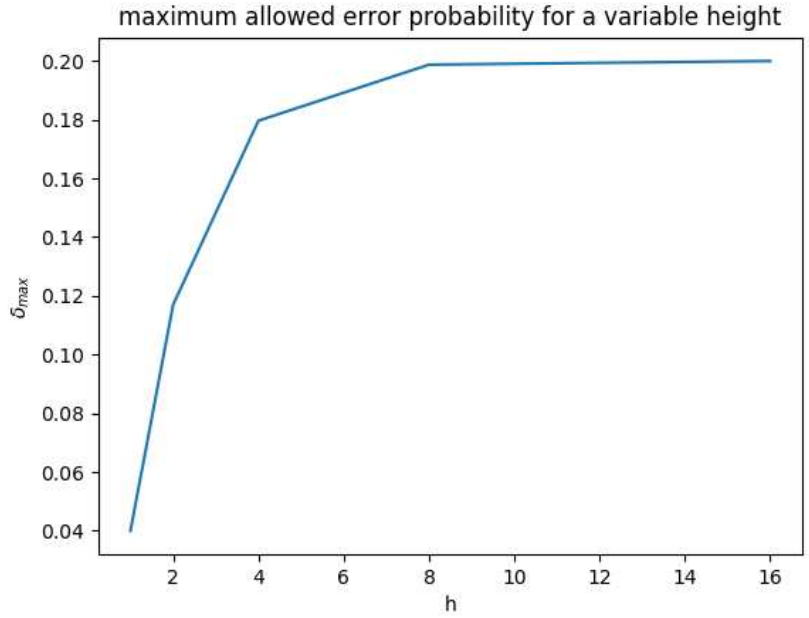


Figure 3.5: This graph shows the value of the maximum allowed value of the confidence parameter given the height value. Note that the confidence is given by $1 - \delta$ and the error probability is given by δ

a variable value of the Radon number. In the setting assumed, the Radon number is dictated by the dimensionality of the data that we wish to learn ($r = d + 2$), the height of Iterated Radon Points algorithm h is dictated by both the data dimensionality d and the number of learners k ($h = \log_r k$). In the plot, we change the data dimensionality and set the number of learners $k = 2000$ and the initial value of the confidence parameter δ_0 to the minimum among the maximum confidence parameter values for all values of r, i.e, $\delta_0 = \min_r\{\delta_{max_r}\}$. We observe that the larger the value of the Radon number r, the lower the improvement rate of the confidence across subsequent iterations. We will expand on that in the experiments section in a machine learning context.

In fig. 3.8, the Radon number r is fixed, and we have a variable value for the height h and accordingly the number of nodes $k = r^h$. Here, it can be observed that the larger the height of the tree, the higher the change rate of the confidence parameter. This is also evident in the general formula obtained for the improved confidence parameter in any synchronization phase (Eq. 3.10). From a geometric perspective, this is also supported by the result obtained from the Iterated Radon Points paper, where the error probability of the computed Radon point with respect to the center point decreases with the height of the tree.

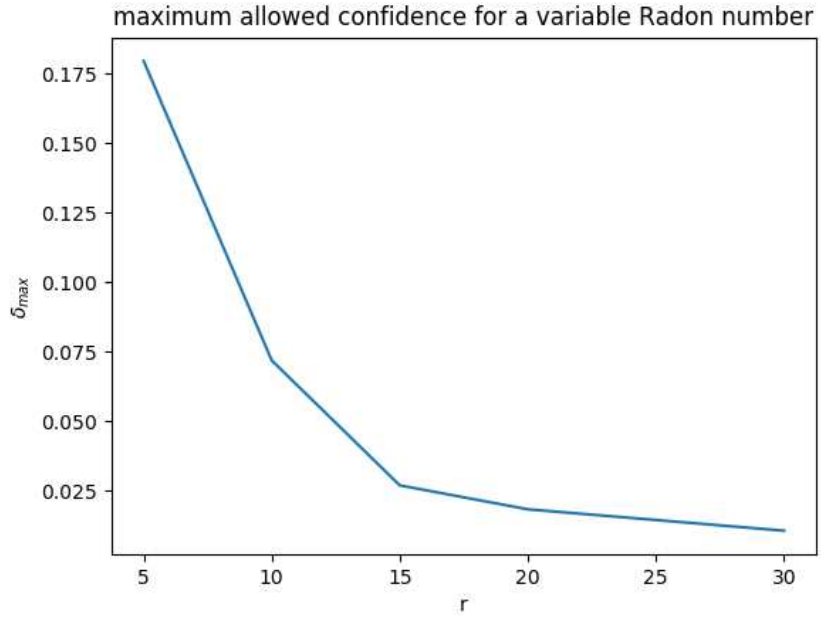


Figure 3.6: This graph shows the value of the maximum allowed value of the confidence parameter given the Radon number

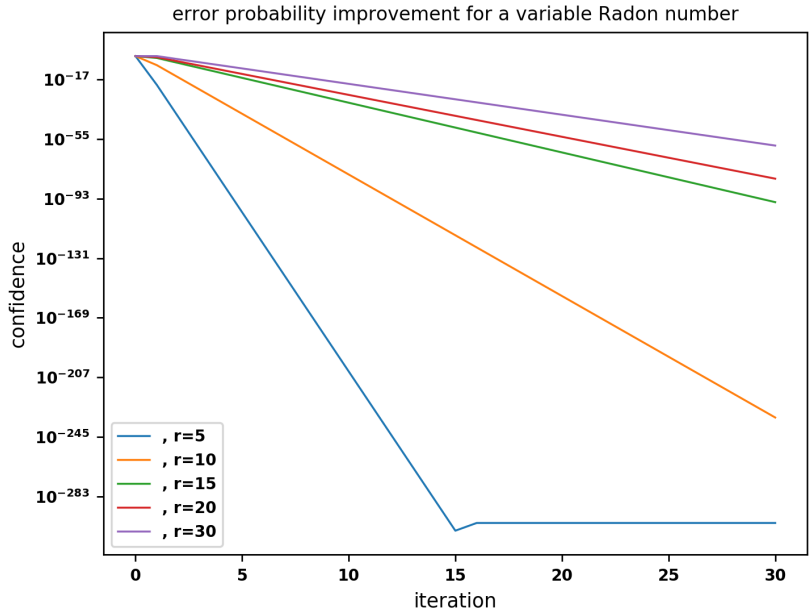


Figure 3.7: This graph shows the improvement of the confidence value across different iterations for different Radon numbers. The height of Iterated Radon Points tree is two in all of them. Since the confidence parameter is $(1-\delta)$, a lower value for δ improves the confidence. (logscale).

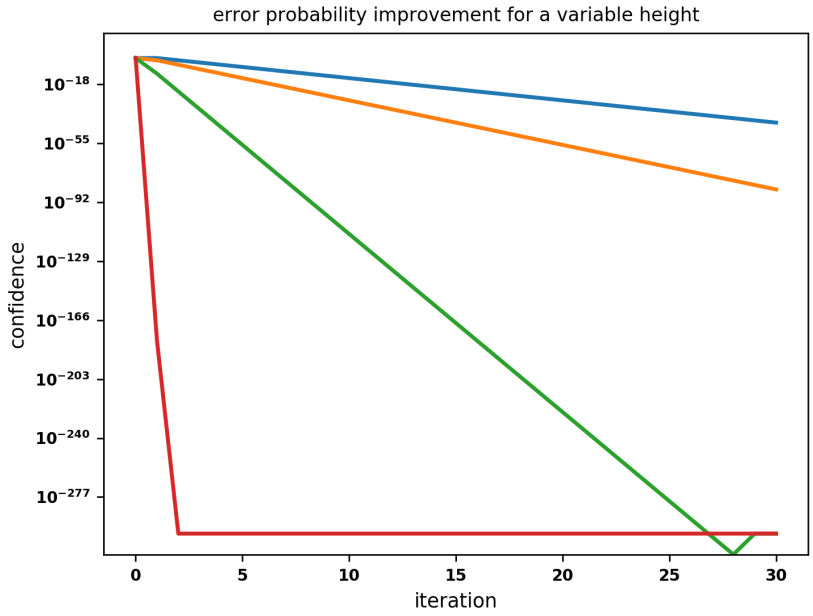


Figure 3.8: The confidence improvement is illustrated for a fixed Radon point($r = 5$) and a changing height. (log scale).

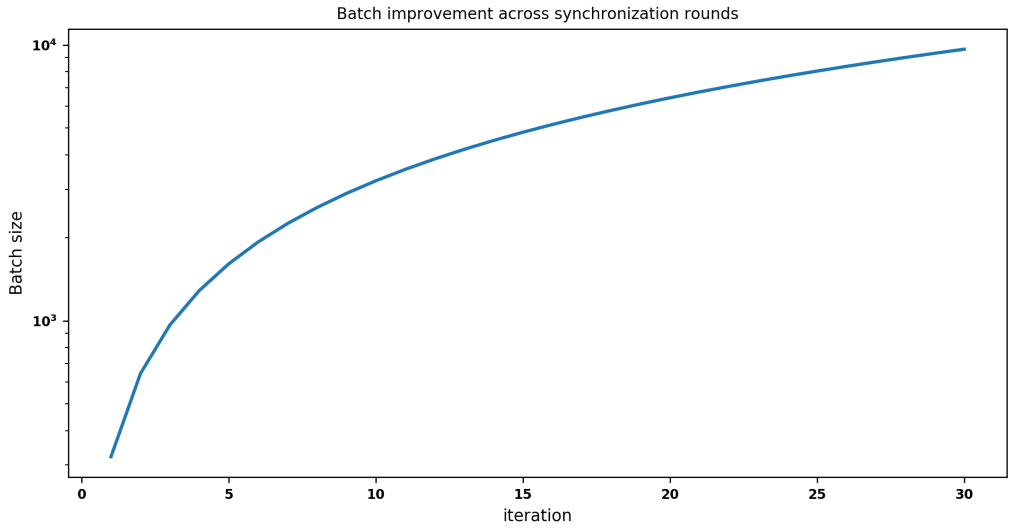


Figure 3.9: The batch improvement across the synchronization rounds. $\delta_0 = 0.04$, was taken to be the maximum allowed as given by Eq. 3.5, the value of the initial batch size is therefore $b_0 = 160$ as given by Eq. 3.11. $h = 2, r = 5, k = 5^2$. For any corresponding improved batch value per learner on the y-axis, the actual number of samples seen by one learner is $b_0 \cdot t$ and by all the learners this value becomes a multiple of k, t denotes an iteration index. (logscale).

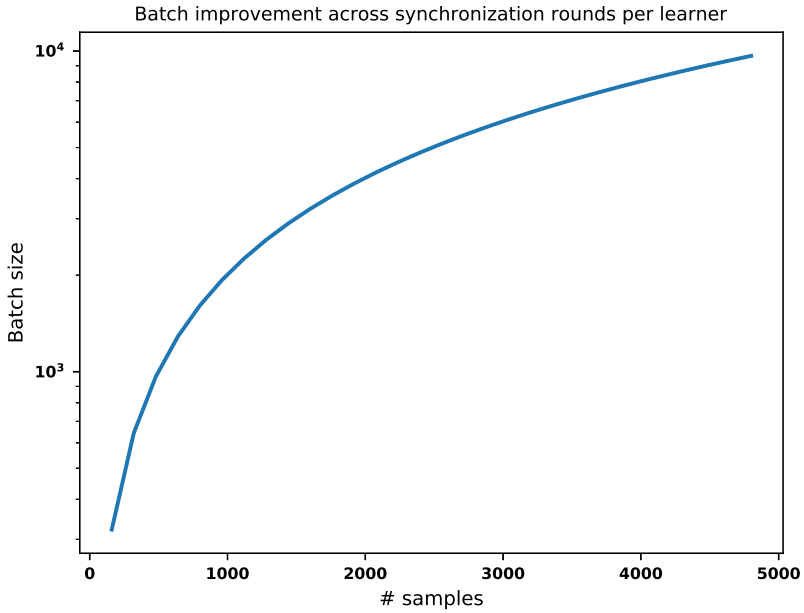


Figure 3.10: The batch improvement across the synchronization rounds, the number of samples on the x-axis indicates the actual number of samples seen by a single learner, which is given by $b_0 \cdot t$. (logscale).

4 Empirical Evaluation

4.1 Setup

The experiments have been executed on a local simulated environment in order to allow identical conditions to be present for the proposed different approaches. In this case, the same samples are observed in the different synchronization approaches thereby preventing performance discrepancies that are not attributed to the approach itself, especially since the computation times vary across these approaches. Moreover, running these experiments in a distributed environment would have affected the results based on the network state.

As mentioned earlier, in this work, a linear model with stochastic gradient descent as the update rule is used in the proposed approach. The learning rate of SGD was determined by a simple adaptation rule given by $\eta = \frac{\eta_0}{1+\frac{const}{\#samples}}$ where the parameters *const* and η_0 had to be fine-tuned.

In most experiments, the proposed approach which employs the Radon point is compared against two other different approaches, namely the average point and the geometric median together with no synchronization at all (no-sync). Based on the literature in the related work section, it is reasonable to assume that the performance of the geometric median should be the most competitive with the Radon point. For implementing the geometric median, the method in Beck and Sabach [2] which is an adaptation of the Weiszfeld algorithm [27] has been used.

In the experiments, use was made of both synthetic and real-world datasets. Generating datasets synthetically allowed us to control certain parameters such as the data dimensionality and to examine their effect on our approach as will be shown later. The real-world dataset is YearPredictionMSD [21]. The data consists of 515345 data samples that belong to songs, 90 real-valued attributes comprising audio features of the songs, and a real valued label indicating the release year of the song, making the task associated with this dataset a regression task. A single learner with a linear model can learn this dataset with a high correct prediction rate (99%). In all the experiments, the square loss function is used. The evaluation criteria used is the cumulative loss of all the learners.
Note that for a d-dimensional dataset, linear regression maintains a linear model

of $d+1$ coefficients given by $\{w_0, \ldots, w_{d+1}\}$ where w_0 is the intercept, therefore making the Radon number $r = d+3$

4.2 Experiments

4.2.1 Evaluation on Uncontaminated data streams

In the first experiment, the algorithm is run on the dataset without perturbing its samples.

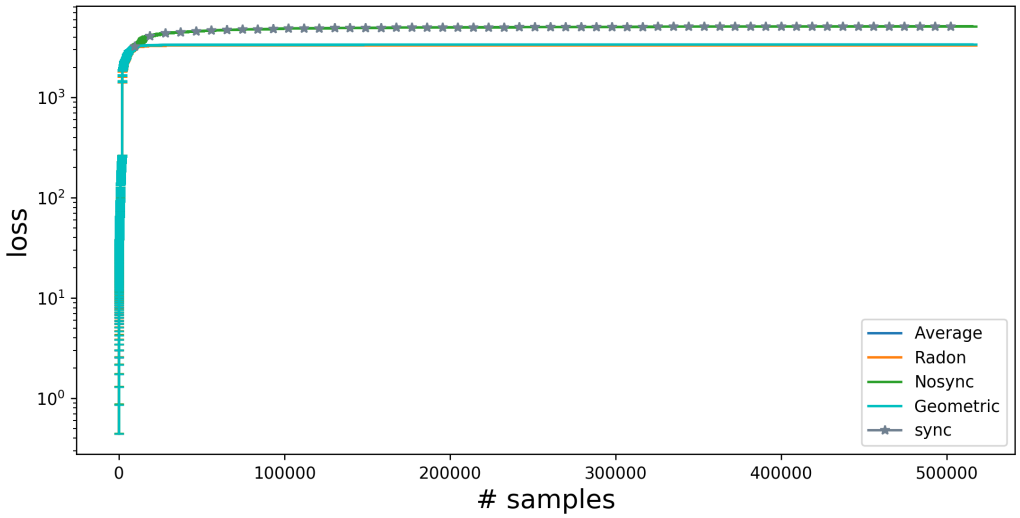


Figure 4.1: Comparing the performance of all three approaches when there is no noise in the data($\sigma = 0$). Here, $b = 100$, $T = 55$, $k = 93$, $h = 1$ (T denotes the number of synchronization rounds). The gray stars on the plot indicate the synchronization points, i.e., the total number of data samples seen by all the learners when a new synchronization phase is initiated. The gray points are plotted on the no-sync curve, the highest curve in this graph but it is not intentional and the idea is just to show the points of the synchronization phases

In fig. 4.1 which depicts the scenario with uncontaminated data, all aggregation approaches perform at least 156% better than no synchronization in this run (in other runs the improvement could reach 500%). Apart from no-sync, the improvement between the aggregation approaches is negligible(less than 2%). The sequence of synchronization points is given by $b \cdot k \cdot t$ where b denotes the batch size, k the number of learners and t the number of the synchronization round. It can be seen that after the first synchronization round, the performance of any synchronization approach diverges from that of no-sync.

4.2.2 Evaluation on noisy data streams

To give motivation for the following experiments, let us first recap what goal needed to be achieved in this research.
This thesis topic has been motivated on two levels, the first was studying the Radon point theoretically and the second examining its robustness empirically. In the experiments, the Radon point is compared with another two aggregation approaches that were already discussed in earlier chapters, namely the average point and the geometric median. The experiments in which the data remained uncontaminated, all three approaches have delivered almost equal performance and are therefore equally robust.

However, in the discussion throughout this paper, it has been argued that noise is bound to occur in the data thereby causing distortion in a local hypothesis in a distributed setting. During synchronization, these distorted hypotheses constitute outlier models and therefore adversely affect the combined global model. It has also been argued that the average point is susceptible to such outliers and draws the average point far from the optimal point, a property against which the Radon point is more immune (see fig. 1.1). A similar argumentation of immunity was also made for the geometric median. To demonstrate the performance of the Radon point in these scenarios, noise is introduced to the data.

Given in what follows is a description of the noise model used in the experiments. It is assumed that the data arrives from a fixed distribution with occasional random occurrence of noise in the data samples. The data samples are perturbed independently using additive Gaussian noise by an adversary and therefore no assumptions are made on the noise magnitude. As mentioned in the introduction, this experiment may model a real-life scenario in which several distributed sensors are placed in an environment to perform a given target task, and where from time to time these sensors malfunction and possibly produce erroneous readings. A failure and recovery probability are assigned to every sensor in the environment such that during failure times, each feature of an incoming sample has a 50% failure probability. A noisy feature is perturbed with an additive Gaussian noise, i.e., $x_i = x_i + \xi$ where $\xi \sim N(0, \sigma)$ for some $\sigma \in \mathbb{R}$ whose value determines the magnitude of the noise. The influence of the standard deviation value will also be examined in the experiments.

In the following experiment depicted by fig. 4.2, the data has been perturbed with $\sigma = 1.5$. A failure probability $p_{failure} = 0.02$ is also assigned, a recovery probability $p_{recovery} = 0.7$ and batch size $b = 100$. The total number of perturbed samples amounted to 14480 ($\sim$ 2.7% of all the data samples), the number of synchronization rounds $T = 55$.

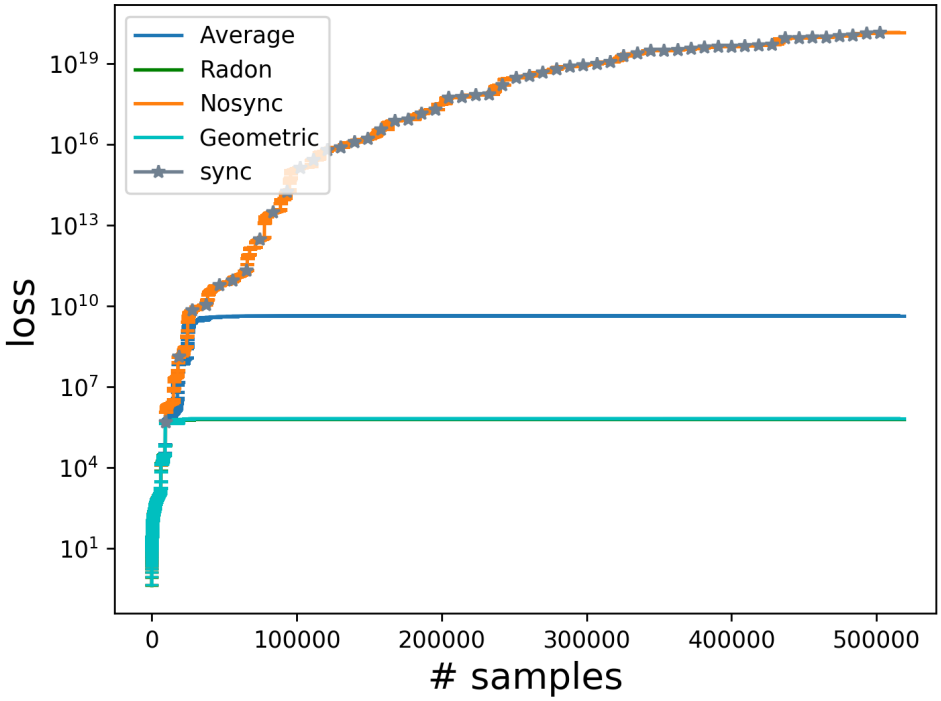


Figure 4.2: Comparing the performance of the different approaches with an additive Gaussian noise, $\sigma = 1.5$(logscale).

Since it is difficult to make out the difference between the geometric median and the Radon point on the plot due to the gap between these two approaches and averaging, the graph is plotted again in fig. 4.3 where only these two approaches are shown.

It can be seen that the gap between averaging and the other two approaches is already considerably increased compared to fig. 4.1, the experiment without noise. It can be seen that the performance of the averaging method deteriorates compared to the previous plot and the Radon point's performance is orders of magnitude better. In this run, Radon also performed better than the geometric median with a 7% improvement and orders of magnitude better than averaging with 10^4%.
In light of what has been said about the geometric median, it can be seen that the performance of Radon point and the geometric median remain close to each other.

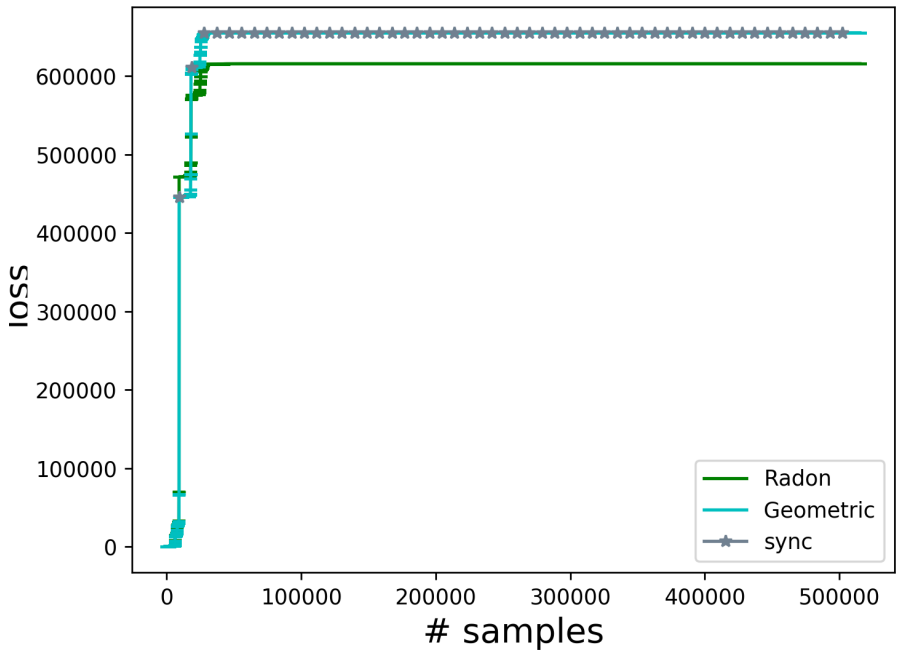


Figure 4.3: The same plot from the previous figure where the loss of the Radon point and geometric median are alone extrapolated the log scale removed

4.2.3 Evaluation of the Radon point versus the Geometric median

In order to get a better insight into how Radon point and the geometric median perform against each other, both of them are run on the dataset for 50-100 times where in each run the order of the samples is randomized in order to get a different ordering. This was done for two experiments where in the first one the standard deviation $\sigma = 1.5$ and in the second $\sigma = 2$

In the following experiment(fig. 4.5), the standard deviation of the Gaussian noise is increased to $\sigma = 2$ and run both approaches for 50 times.

4.2.4 Complexity and Runtimes

The complexity of the aggregation is another critical measure in the synchronization scheme proposed in this work, especially in light of the problem illustrated in fig. 3.1. In my experiments on Apache Storm, which is a realtime distributed computation framework, when the batch size was 100 and the throughput 100k samples/second, using the average point, $\sim$ 45% of the data was discarded and

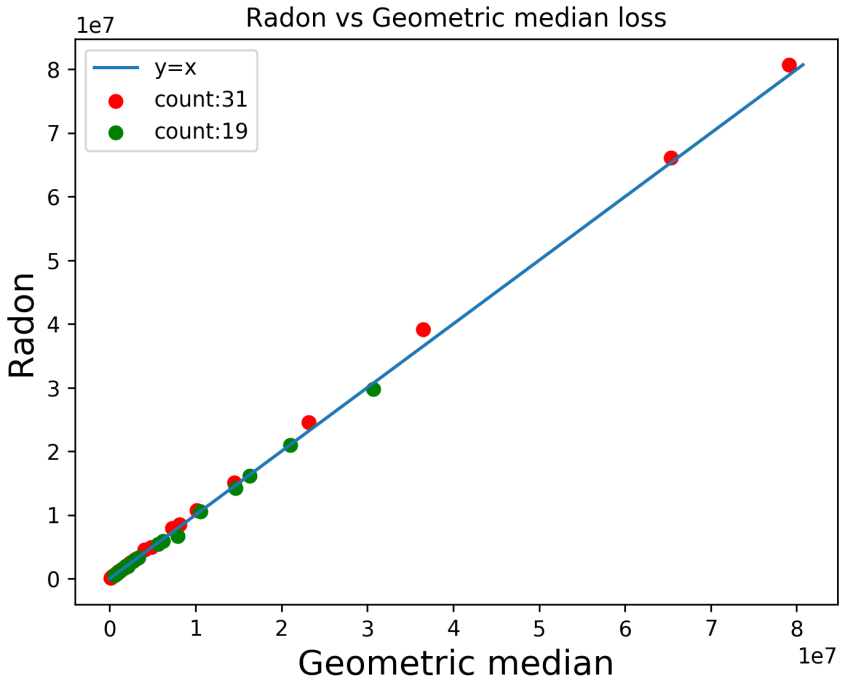


Figure 4.4: For this experiment, both approaches are executed on the dataset for 98 times where each time both algorithms were executed with the same ordering of the samples. The blue line represents all the points where the loss of the geometric median equals the loss of the Radon points. The graph shows how many times the Radon point approach outperforms the geometric median approach. In other words, the points below the line indicate a better performance for Radon (colored by the green dots) and the points above the line indicate better performance for the geometric median (colored by red). It can be seen that the Radon point outperforms the geometric median 40 times whereas the Geometric median outperforms the Radon point 58 times, i.e., it performs better 145% of the time. $\sigma = 1.5$

using the Radon point $\sim 60\%$ of the data was discarded. In essence, this would render a computationally expensive synchronization approach that is deployed on a high rate data streams futile. If ,however, the rate of the data streams is not too high, then synchronization would be beneficial. This obstacle has urged us to compare the different synchronization approaches in a controlled local environment where the results are independent of problems that are associated with the environment or the network. It has been shown that the complexity of this synchronization approach is equal to $d^2 \cdot k$, d is the data dimensionality and k the number of models. We just briefly mention that the implementation for finding the geometric median is based on the paper Beck and Sabach [3], in which an improved

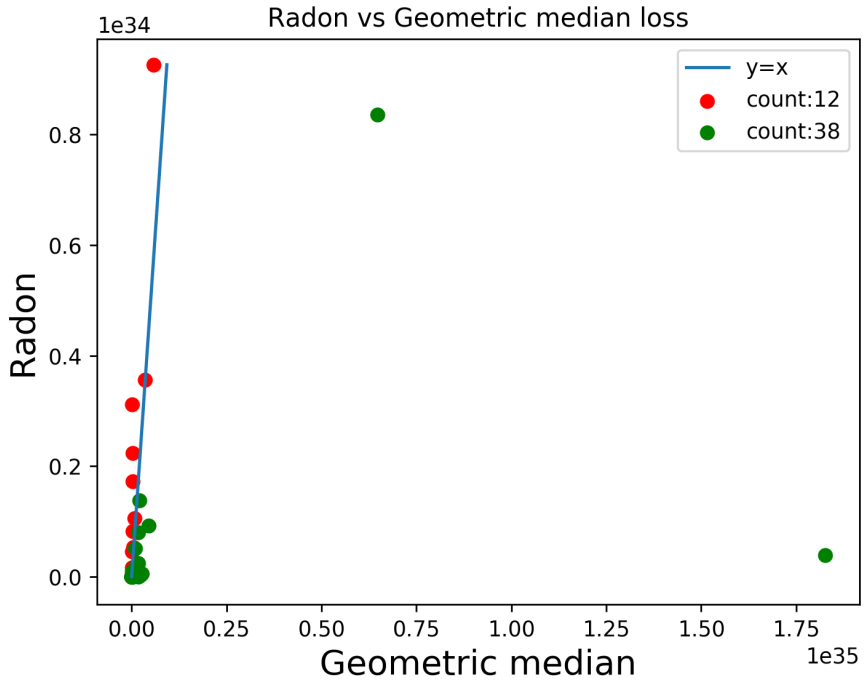


Figure 4.5: Here, the Radon point beats the geometric median 38 times whereas the geometric median beats the Radon point 12 times, i.e., the Radon point performs better 316% of the time. $\sigma = 2$

initialization method is proposed and accordingly a sublinear rate of convergence of the method is proved.

Now let us compare the runtime of each approach in practice.

4.2.5 Dependence on parameters

The influence of the standard deviation on the performance gap is now further examined between the Radon point and the other approaches. With the magnitude of noise in fig.4.7, synchronizing by averaging becomes so detrimental to the models that no-sync becomes more beneficial. The gap between the geometric point and the Radon point also becomes noticeably larger with an improvement ratio of 150% for the Radon point.

In the theoretical part, irrespective of the presence of noise, it has been shown that the error probability exponentially improves across the synchronization rounds. In the experiment in fig. 4.8, the loss decay per point is examined rather than the cumulative loss as a result of synchronization.

In the experiment for the plot in fig. 4.11, the influence of the failure probability

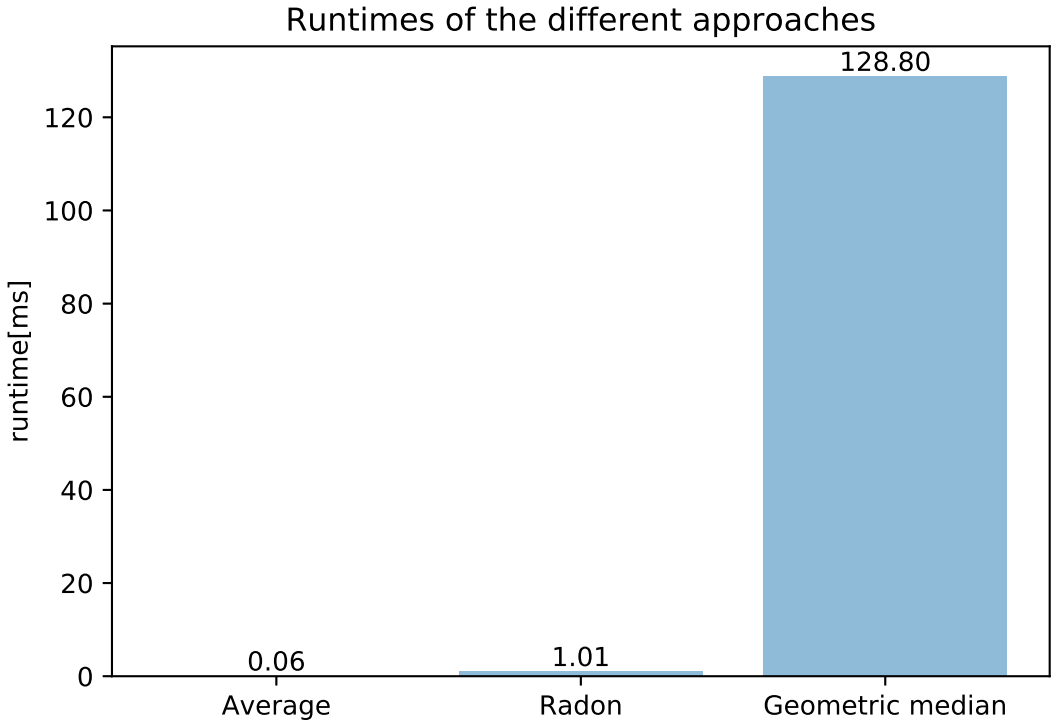


Figure 4.6: The runtimes are averaged over 10 different runs on the dataset YearPredictionMSD where each run had 55 synchronization rounds. While averaging is 18 time faster than the Radon point, The Radon point is 128 times faster than the geometric median.

on the loss ratio between the Radon point and the other approaches is tested. Note the a lower loss ratio indicates an improvement for the Radon point.

In fig. 4.11 and 4.12, it can be seen that in general as the failure probability increases, the loss ratio between the Radon point and the other approaches decreases. This behavior can be explained by the fact that the more noisy samples are observed in the data, the more distorted the output model of the aggregation approaches will be. Since the ratio between the Radon point and the geometric median manifests itself with a larger magnitude of noise, a separate experiment which the standard deviation is increased to $\sigma = 2$ is performed. Contrary to the other approaches, when the failure probability approaches 0.85, the ratio between the Radon point and geometric median increases thereby indicating a break point for the Radon point against the geometric median.

Likewise, fig. 4.13, the influence of the recovery probability is tested on the loss ratio between the Radon point and the other approaches.

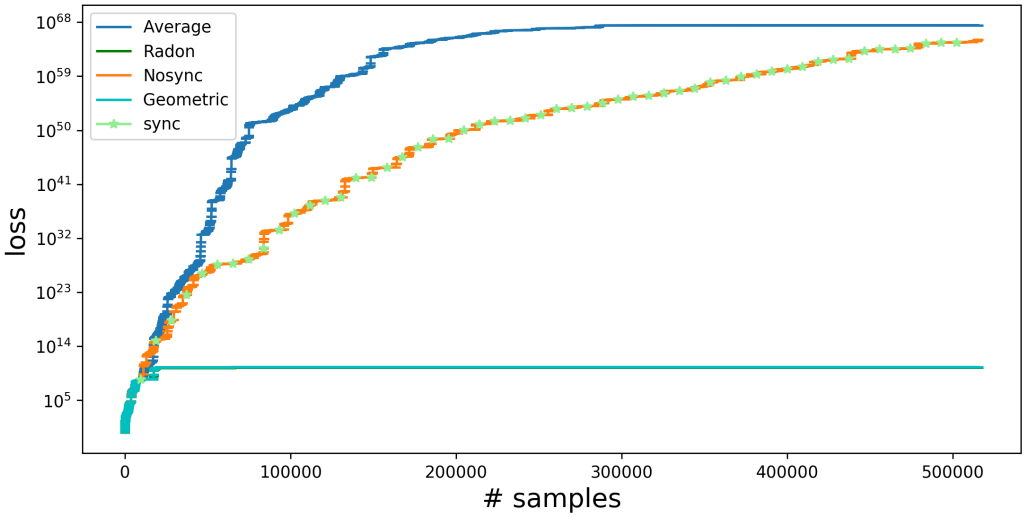


Figure 4.7: $\sigma = 2$, Radon performs 119% better than the geometric median(log scale).

In fig. 4.13, it can be observed that the loss ratio between the Radon point and the other approaches is a monotonically increasing function. This confirms the result obtained in the previous plot with the failure probability. An increasing recovery probability indicates that the occurrence of outliers is becoming rarer. This indicates that the models are being deformed less and less which brings the performance gap between the Radon point and the other approaches closer together. The ratio is computed by averaging over 10 runs for each recovery probability.

It can be observed that for the same variance value, increasing the dimensionality of the set increases the ratio of improvement between the Radon point and the other approaches. This can be explained by the fact that there are more perturbed attributes and therefore the influence of the noise on the learned model is more significant. For the experiment in fig. 4.14, datasets of different dimensions with an additive Gaussian noise have been generated.

The batch size is a critical parameter that affects the amount of communication in a distributed learning algorithm. A small batch size entails frequent communication between the learners thereby consuming much of the network bandwidth. This also aggravates the synchronization overhead, a problem which was explained earlier in fig. 3.1. On the other hand, a large batch size results in diminished improvement in the overall objective loss function, since eventually, a degree of information exchange is desirable. The same experiment is repeated with the average

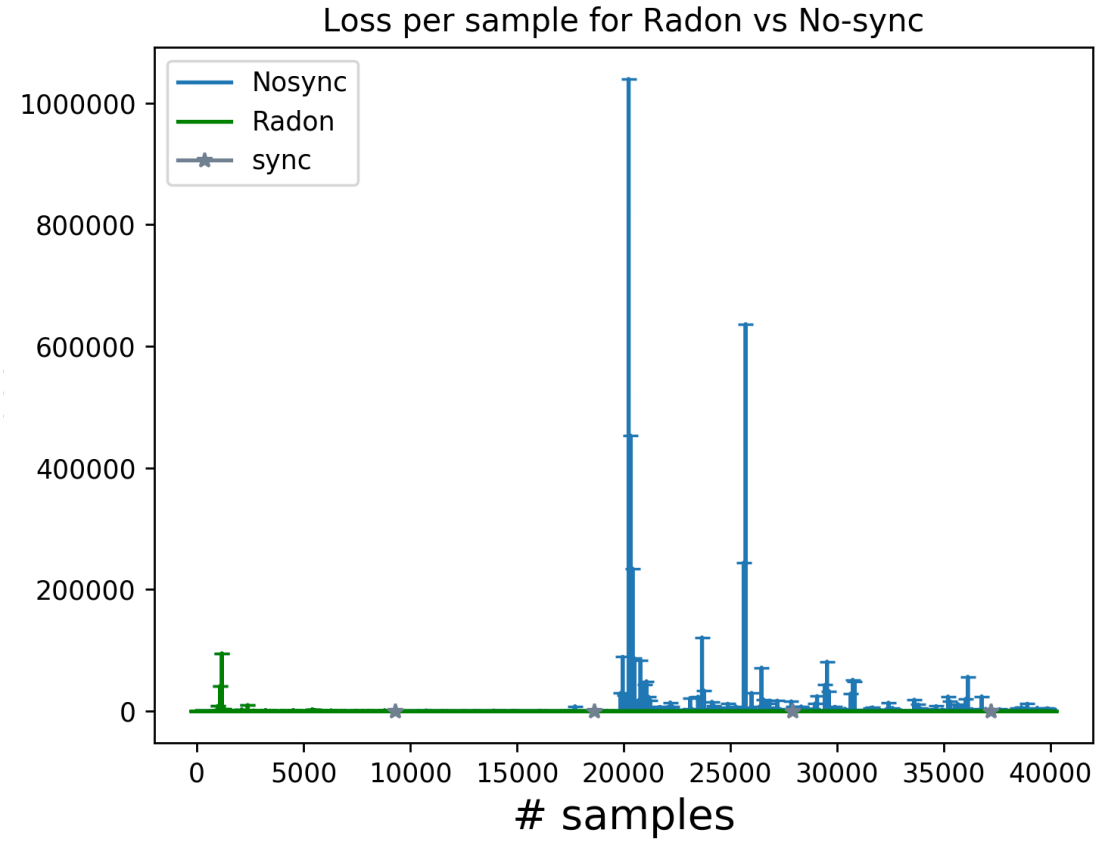


Figure 4.8: In this plot the Radon approach is compared with no-sync. It can be seen that no-sync continues to have spikes for the loss value whereas after synchronization the spikes of the Radon approach are diminished after each synchronization.

point but with a smaller variance value.

In fig. 4.16, it can be seen that increasing the batch size decreases the loss ratio between Radon and the Geometric median, indicating that increasing the batch size has less influence on Radon point than the Geometric median. This can be explained by the fact that a larger batch size indicates that a broken sensor will learn longer with a deformed model and suffer more loss until it synchronizes. However, since the Radon point is more robust to strong outliers, the overall combined model is not influenced as much as the Radon point. This has a paramount connotation in terms of the amount of communication required. In general, there is a trade-off of model quality versus the communication amount and if the scenario requires saving communication then the Radon point will be less influenced.

Up until now, the height of the Iterated Radon Points tree has been equal to one,

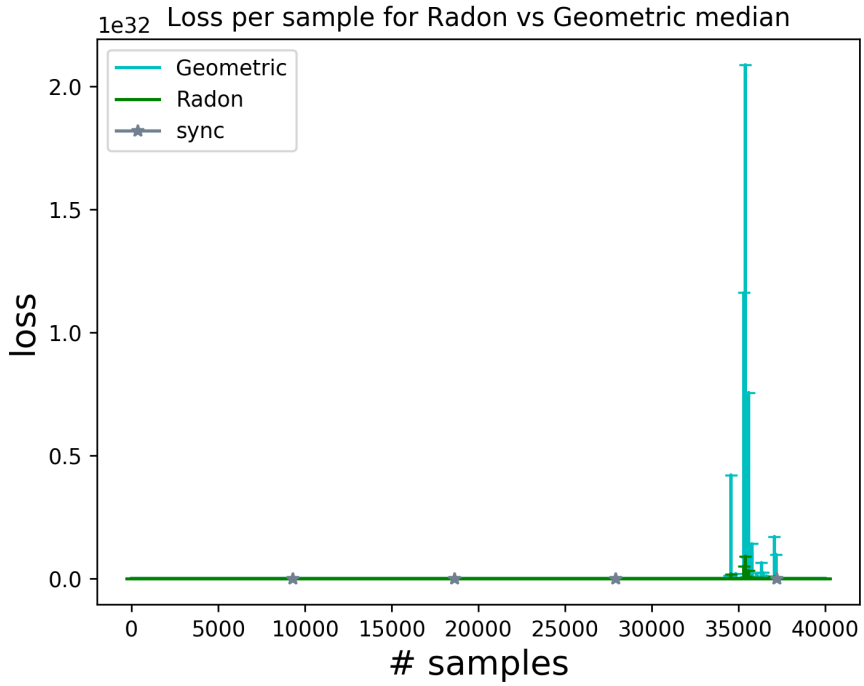


Figure 4.9: Here some perturbation in can be discerned in the noise value before the fourth synchronization round (the fourth gray star), indicating a sensor failure event that gave rise to some outliers models. The Radon point model is less influenced by this failure since its loss spike is low in comparison to the geometric median spike. This complies with what has already been shown in the previous experiments where the Radon point was more robust than the geometric median with this magnitude of noise. $\sigma = 10$

indicating that the number of nodes was equal to the Radon number. Now let us proceed to examine the influence of selecting an aggregated model from different levels of the tree. In Clarkson et al. [7], it has been shown that the error probability of the Iterated Radon Points algorithm is doubly exponentially small with respect to the tree height. Now let us examine this hypothesis empirically in the context of learning, namely, if the tree height is given by h_{max}, then a Radon point is selected uniformly at random from the points at level $\{0, 1, \ldots, h_{max}\}$. Since the learners set size grows exponentially with the height, a large dataset was needed in order to have a sufficient number of samples to allow for several synchronization rounds to occur. The dataset in this experiment was generated synthetically and contains $5M$ 5-dimensional samples. The program is executed up until the maximum height that will still allow enough synchronization rounds in the system. In this case, the number of nodes $r^h = (d+2)^h$ would be the set $k = \{8, 8^2, 8^3, 8^4\}$. The batch size is

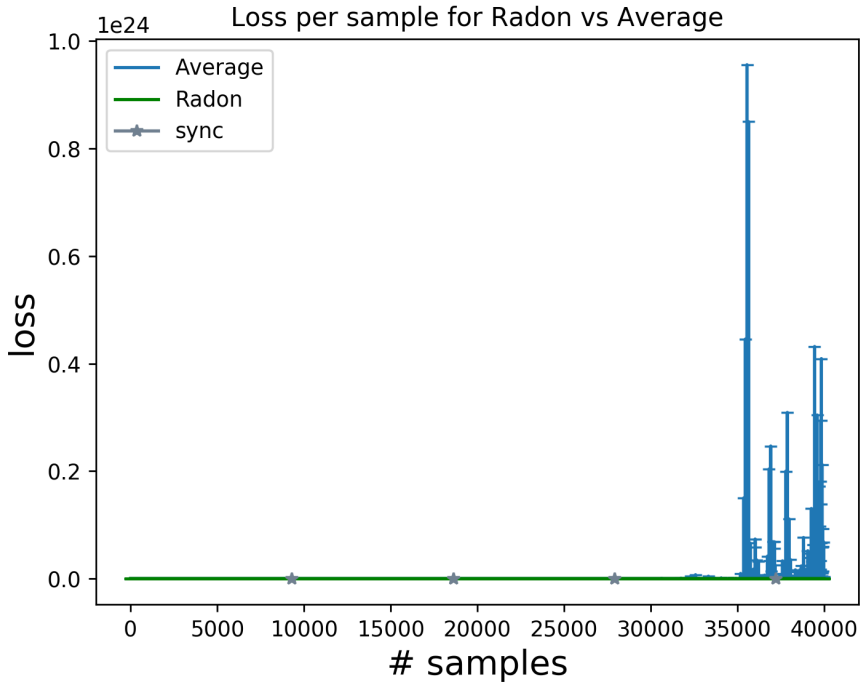


Figure 4.10: The same explanation of the geometric point applies to the average point as well, only with averaging it suffices to have a small additive noise in order to distort the averaged model. Clearly, synchronization by averaging did not improve the loss of the averaged model here, but rather degraded the model since the loss spike after synchronization is higher than before.$\sigma = 2$

also set to $b = 100$, allowing for $T = |S|/(k \cdot b) = 2M/(b \cdot k) = 12$ synchronization rounds in total. This experiment is executed in two different scenarios, one is without perturbing the data and one with perturbing the data. As expected, the loss value decreases with increasing the height, and in the presence of noise, the improvement is more noticeable.

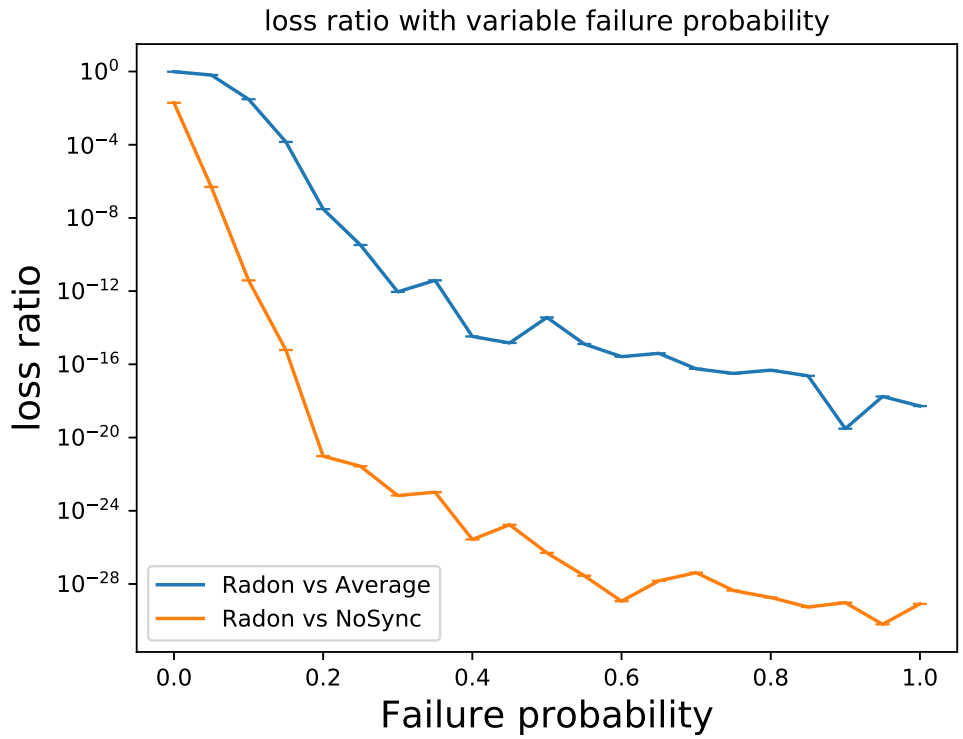


Figure 4.11: Loss ratio between the Radon point and the average point, and the Radon and no-sync vs failure probability. The recovery probability $r = 0.6$ for all the nodes and $\sigma = 1$. The ratio is computed by averaging over 10 runs for each failure probability.(log scale).

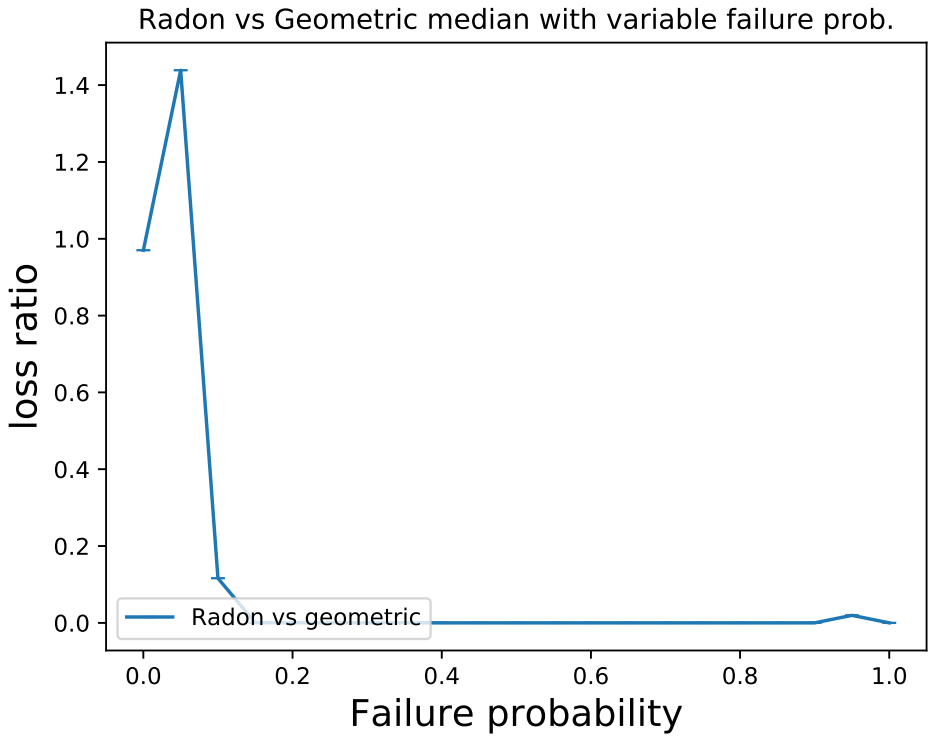


Figure 4.12: Loss ratio between the Radon point and the geometric median versus the failure probability. $\sigma = 2$.

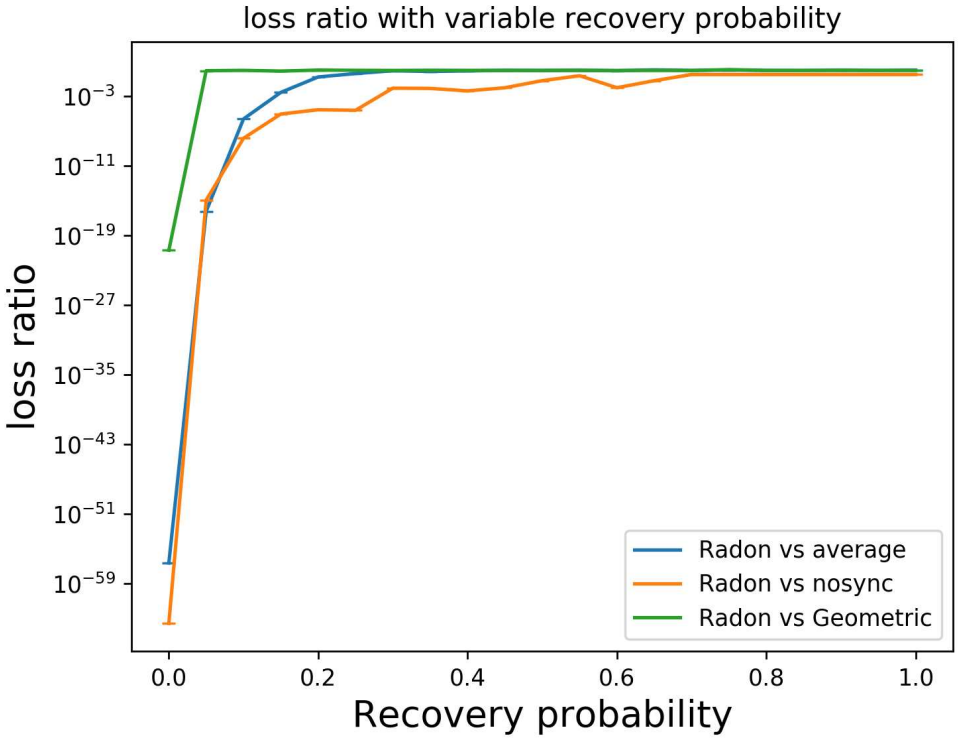


Figure 4.13: Loss versus Recovery probability. failure probability=0.01 for all the nodes. (log scale).

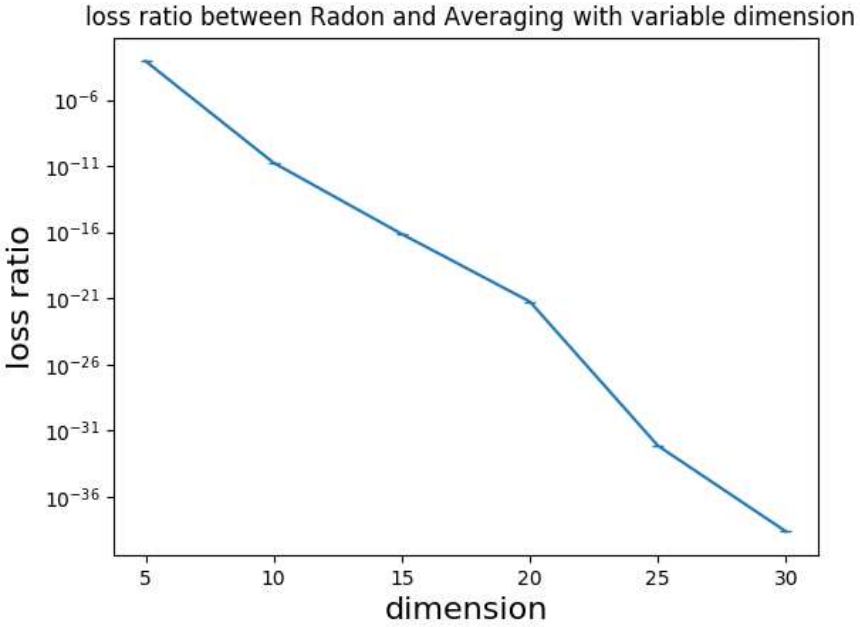


Figure 4.14: The ratio is computed by averaging over 10 runs for each dimension. $\sigma = 8$.

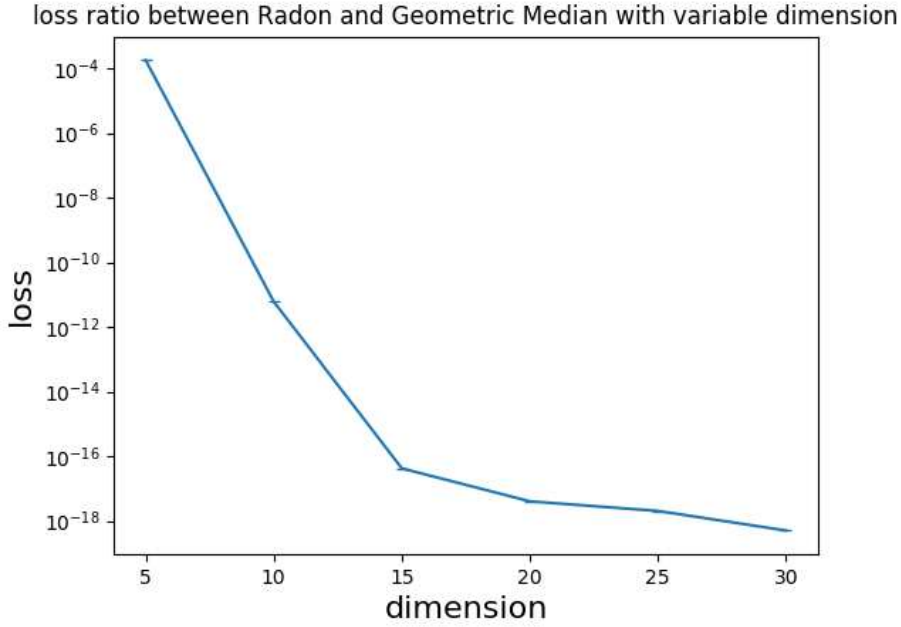


Figure 4.15: The ratio is computed by averaging over 10 runs for each dimension. $\sigma = 12$.

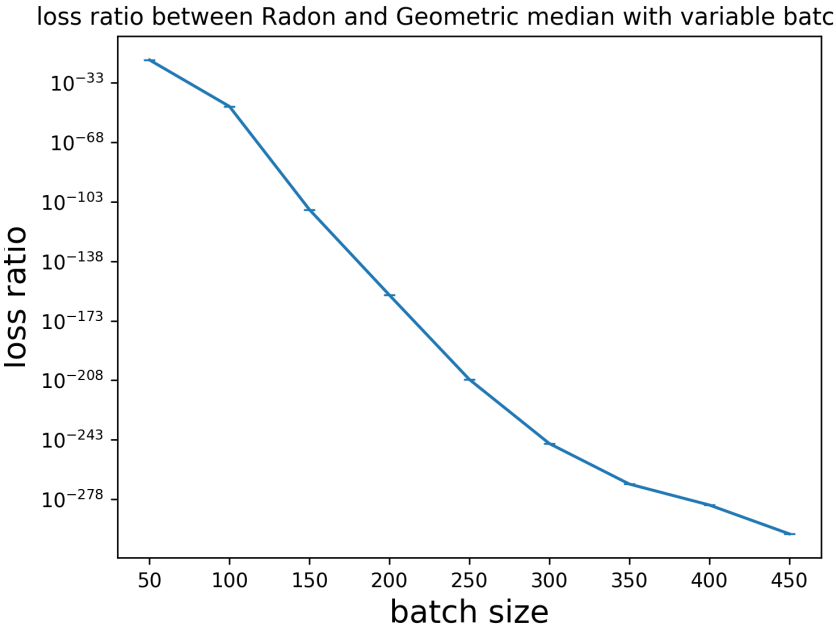


Figure 4.16: $\sigma = 12.5$. The ratio is computed from averaging over 10 runs for each batch size with increments of 50. The dataset in this experiment is synthetic and of dimension $d = 10$ and size $1M$. (log scale)

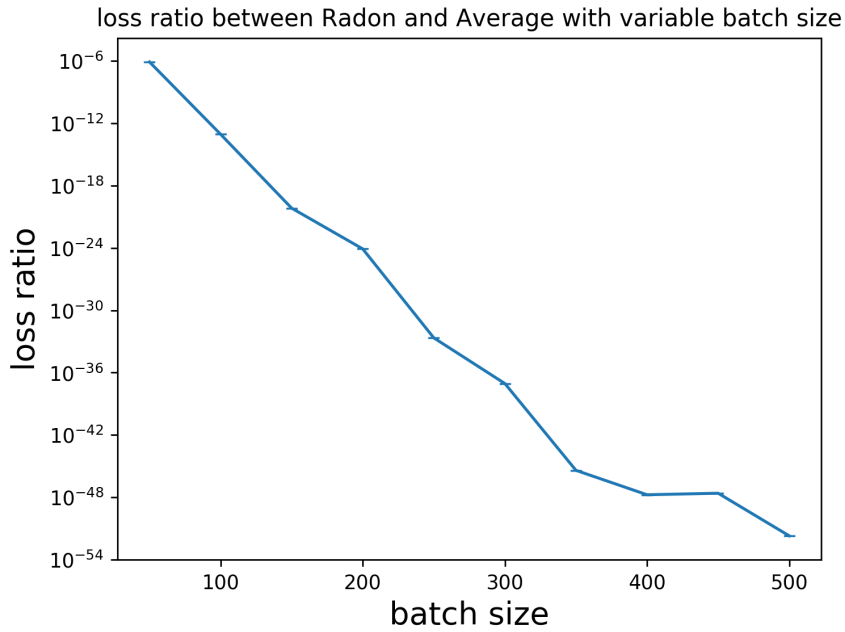


Figure 4.17: $\sigma = 2.5$. (log scale)

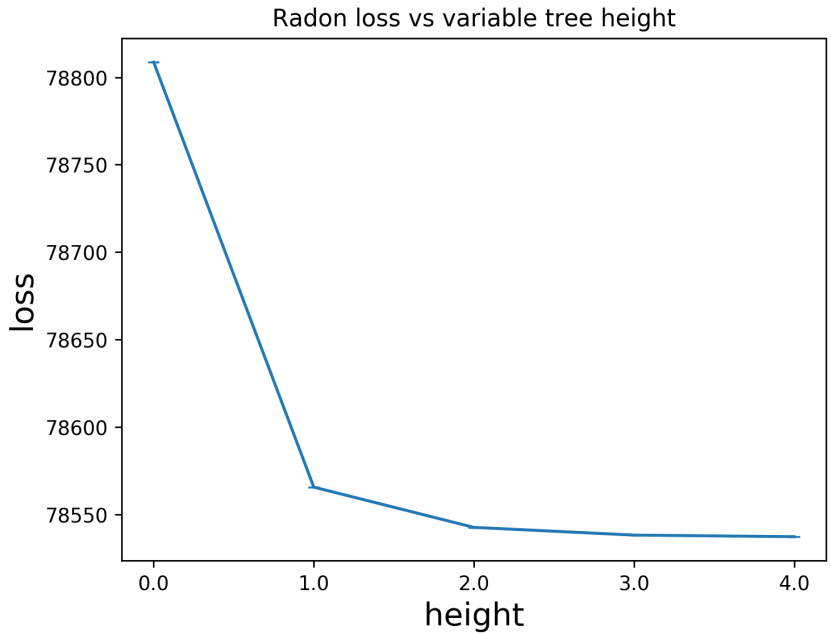


Figure 4.18: In this experiment, the program is executed for 10 times for each tree level value and averaged the loss value over these runs. The data is unperturbed. It can be seen that already starting from height two, the difference in the cumulative becomes negligible, indicating that one can doesn't have to iterate to the full height and therefore reducing the runtime of the algorithm.

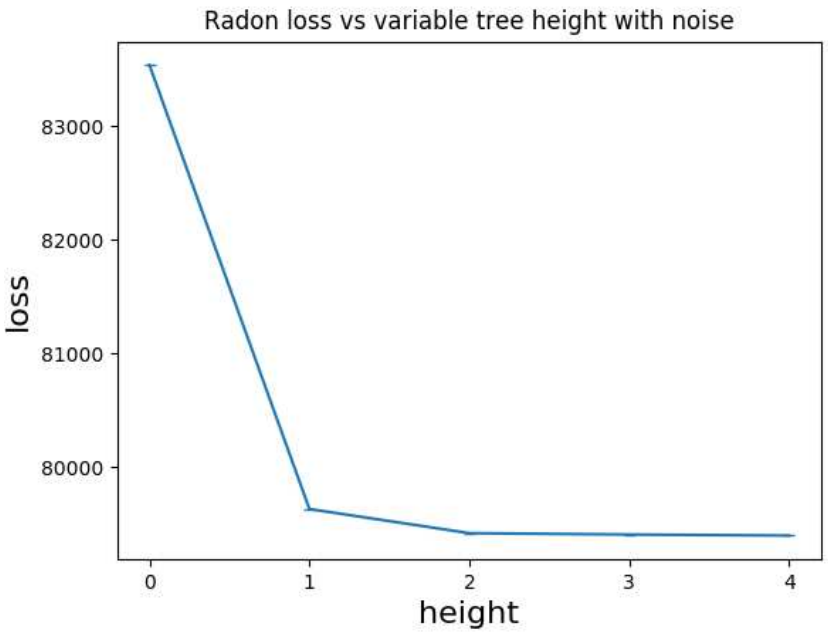


Figure 4.19: The data is perturbed, $\sigma = 2$

5 Discussion, Conclusion and Future Work

5.1 Discussion

We have seen that when the magnitude of the noise increases in the data, the resulting outlier models, i.e., those which deviate from the true model, become stronger, which in turn impacts a model aggregated by averaging to a degree that aggregation could become detrimental to the overall learning system.

Through the experiments, it was shown that when the noise is unbounded, the proposed approach performs better than the geometric median, which was the more competitive approach between the ones that have been examined in this work. This can be explained by the fact that the geometric median, like averaging, is still defined with respect to the distances of the points rather than the Tukey depth. In particular, while the geometric median minimizes the sum of distances, averaging minimizes the sum of squared distances. The Tukey depth of a point, however, is defined with respect to the minimum number of points on one side of the plane going through this point, and Radon point is defined such that it balances the number of points on each side of the plane. Strong outliers, therefore, influence the geometric median more than they do a Radon point although not to the same degree as averaging.

The Radon point, though attractive for scenarios with noisy data, still has some constraints and disadvantages. One constraint is a result of Radon theorem which states that at least $d + 2$ hypotheses are required in order for the Radon point to exist. Furthermore, if the Iterated Radon Points algorithm is applied with a requirement for a certain height, then at least $(d + 2)^h$ different hypotheses are required. This may render the proposed method irrelevant if the height value is large or if the dataset is high dimensional since the number of learners k is not necessarily that large in a real-world setting.

In this work, guarantees with respect to the Radon point for convex optimization have been shown, however, such guarantees are not provided for non-convex optimization and therefore this excludes the utilization of the proposed approach

in neural networks, in which the non-convexity stems from the non-linearity in the activation functions.

Since communication is constrained by the network bandwidth, it is imperative to keep the amount of exchanged data small, thereby indicating that the hypotheses should have a small finite dimension. For the kernel method therefore, the proposed approach would become infeasible since then the dimension depends linearly on the data size.

We have also seen that the complexity of calculating a Radon point is higher than averaging, leading to many data samples being discarded during synchronization. However, when the data is noisy and there is no assumption on the noise magnitude, the Radon point's performance is magnitudes better than averaging and since its computation is faster than the geometric median, it is the winner method for this scenario.

5.2 Conclusion

We have argued for the potential of a Radon point throughout this thesis. A PAC guarantee was provided for the Radon point in a distributed online setting. The performance of the Radon point against other aggregation methods was further demonstrated in practice. The experiments performed showed that in the presence of unbounded outliers in the data, the proposed approach performed considerably better than the the other methods. Indeed, Radon point proved itself as a robust classifier in practice.

In the absence of outliers, a Radon point's performance was comparable to that of averaging. However, since a Radon point is computationally more complex to calculate, depending on the scenario and the trade-off between computational complexity and the learning quality, averaging could be a better option. Likewise, when the noise magnitude is bounded in the data samples, the performance of the Radon point and the geometric median are comparable. With that in mind, computing the geometric median was more complex and although the iterative algorithm used in the experiments [3] converged most of the time, oftentimes the runtime is not fixed and is rather dependent on the number of iterations. In the experiments performed, computing the geometric median point took longer than computing the Radon point, whose computation complexity is fixed regardless of the instance.

5.3 Future work

The theoretical guarantee that has been obtained for the proposed approach applies to data arriving from a fixed distribution. Although the performance of the Radon point did not deteriorate significantly compared to the other examined approaches when there are outliers in the data, it would be interesting to provide a guarantee on its performance or deviation from the optimal model when there is a constant fraction of noisy data in any mini-batch.

The proposed approach relies on a static synchronization scheme. In contrast to a static scheme, dynamic synchronization (Kamp et al. [19]) reduces the amount of communication by allowing the learners to initiate synchronization only when the local models diverge from a global reference model rather than after a fixed mini-batch size. Their synchronization scheme relies on aggregation by averaging, it would therefore be interesting to study how the proposed approach can be adapted to fit a dynamic synchronization scheme and then revise the PAC guarantee proof and examine whether similar probabilistic guarantees can still be maintained.

6 Acknowledgements

Now that I have concluded the rigorous writing of this scientific research, it is time to extend my gratitude and give credit to the people who have made its completion possible.

To my supervisor Michael Kamp whose vast knowledge and continuous guidance have helped make this thesis a success and its results fruitful. In particular as well for conceiving the approach of the Radon point method in batch learning, which paved the way for this research. Not least for his criticism of my initial not so scientific writing, which helped me learn proper scientific work and writing, a process which started with a degree seminar and the expose' of this thesis. It had made getting it right an arduous task that in the end paid off immensely and made the effort well worth it.

To Professor Stefan Wrobel, head of the IAIS Fraunhofer institute and lecturer of the machine learning course, for piquing my interest in the field through his exciting lectures and allowing me to write a master thesis under his supervision. To Dr. Michael Mock for giving me the opportunity to work in the Ferari project, in which I gained experience and insightful critique into the applied aspect of this thesis.

To my parents who have accompanied me throughout this challenging journey, for their moral support from overseas, for their encouragement and patient waiting for the conclusion of this Master's degree.

To my learned friend Linara Adylova for her reading of the thesis and provision of valuable feedback from the reader's perspective.

7 Appendix

Bibliography

[1] Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization, 2015.

[2] Amir Beck and Shoham Sabach. Weiszfeld's method: Old and new results. *Journal of Optimization Theory and Applications*, 164(1):1–40, 2015. ISSN 1573-2878. doi: 10.1007/s10957-014-0586-7. URL `http://dx.doi.org/10.1007/s10957-014-0586-7`.

[3] Amir Beck and Shoham Sabach. Weiszfeld's method: Old and new results. *J. Optim. Theory Appl.*, 164(1):1–40, January 2015. ISSN 0022-3239. doi: 10.1007/s10957-014-0586-7. URL `http://dx.doi.org/10.1007/s10957-014-0586-7`.

[4] Hedde HWJ Bosman, Giovanni Iacca, Arturo Tejada, Heinrich J Wörtche, and Antonio Liotta. Spatial anomaly detection in sensor networks using neighborhood information. *Information Fusion*, 33:41–56, 2017.

[5] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[6] Nicolò Cesa-Bianchi, Shai Shalev-Shwartz, and Ohad Shamir. Online learning of noisy data with kernels. *CoRR*, abs/1005.2296, 2010. URL `http://arxiv.org/abs/1005.2296`.

[7] K. L. Clarkson, David Eppstein, Gary L. Miller, Carl Sturtivant, and Shang-Hua Teng. Approximating center points with iterated radon points. In *Proceedings of the Ninth Annual Symposium on Computational Geometry*, SCG '93, pages 91–98, New York, NY, USA, 1993. ACM. ISBN 0-89791-582-8. doi: 10.1145/160985.161004. URL `http://doi.acm.org/10.1145/160985.161004`.

[8] Riccardo Crepaldi, Mirko Montanari, Indranil Gupta, and Robin H Kravets. Using failure models for controlling data availability in wireless sensor networks. In *INFOCOM 2009, IEEE*, pages 2786–2790. IEEE, 2009.

[9] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *CoRR*, abs/1012.1367, 2010. URL `http://arxiv.org/abs/1012.1367`.

[10] David L. Donoho and Miriam Gasko. Breakdown properties of location estimates based on halfspace depth and projected outlyingness. *Ann. Statist.*, 20(4):1803–1827, 12 1992. doi: 10.1214/aos/1176348890. URL `http://dx.doi.org/10.1214/aos/1176348890`.

[11] Jiashi Feng, Huan Xu, and Shie Mannor. Outlier robust online learning. *CoRR*, abs/1701.00251, 2017. URL `http://arxiv.org/abs/1701.00251`.

[12] Lisa Frye, Liang Cheng, Shenfu Du, and Michael W Bigrigg. Topology maintenance of wireless sensor networks in node failure-prone environments. In *Networking, Sensing and Control, 2006. ICNSC'06. Proceedings of the 2006 IEEE International Conference on*, pages 886–891. IEEE, 2006.

[13] Joao Gama and Mohamed Medhat Gaber. *Learning from Data Streams: Processing Techniques in Sensor Networks*. 1 edition. ISBN 3540736786, 9783540736783.

[14] Thomas Gärtner and Olana Missura. Online optimisation in convexity spaces. In *Proceedings of the NIPS Workshop on Discrete and Combinatorial Problems in Machine Learning (DISCML)*, 2014.

[15] Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. *Bayes and Tukey Meet at the Center Point*, pages 549–563. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-27819-1. doi: 10.1007/978-3-540-27819-1_38. URL `http://dx.doi.org/10.1007/978-3-540-27819-1_38`.

[16] Ralf Herbrich, Thore Graepel, and Colin Campbell. Bayes point machines. *J. Mach. Learn. Res.*, 1:245–279, September 2001. ISSN 1532-4435. doi: 10.1162/153244301753683717. URL `http://dx.doi.org/10.1162/153244301753683717`.

[17] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[18] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 315–323.

Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/4937-accelerating-stochastic-gradient-descent-using-predictive-variance-reductio pdf.

[19] Michael Kamp, Mario Boley, Daniel Keren, Assaf Schuster, and Izchak Sharfman. *Communication-Efficient Distributed Online Prediction by Dynamic Model Synchronization*, pages 623–639. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-662-44848-9. doi: 10.1007/978-3-662-44848-9_40. URL http://dx.doi.org/10.1007/978-3-662-44848-9_40.

[20] Michael Kamp, Mario Boley, and Thomas Gärtner. Parallelizing randomized convex optimization. In *Proceedings of the 8th NIPS Workshop on Optimization for Machine Learning*, 2015.

[21] M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

[22] Ryan Mcdonald, Mehryar Mohri, Nathan Silberman, Dan Walker, and Gideon S. Mann. Efficient large-scale distributed training of conditional maximum entropy models. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1231–1239. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3881-efficient-large-scale-distributed-training-of-conditional-maximum-entropy-m pdf.

[23] Johann Radon. Mengen konvexer Körper, die einen gemeinsamen Punkt enthalten. *Mathematische Annalen*, 83(1-2):113–115, 1921. doi: 10.1007/bf01464231. URL http://dx.doi.org/10.1007/bf01464231.

[24] Jonathan Rosenblatt and Boaz Nadler. On the optimality of averaging in distributed statistical learning. 2014. doi: 10.1093/imaiai/iaw013.

[25] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984. ISSN 0001-0782. doi: 10.1145/1968.1972. URL http://doi.acm.org/10.1145/1968.1972.

[26] N Vijayakumar and Beth Plale. Prediction of missing events in sensor data streams using kalman filters. In *Proceedings of the 1st Int'l Workshop on Knowledge Discovery from Sensor Data, in conjunction with ACM 13th Int'l Conference on Knowledge Discovery and Data Mining*, 2007.

[27] E. Weiszfeld and Frank Plastria. On the point for which the sum of the distances to n given points is minimum. *Annals of Operations Research*, 167 (1):7–41, 2009. ISSN 1572-9338. doi: 10.1007/s10479-008-0352-z. URL `http://dx.doi.org/10.1007/s10479-008-0352-z`.