Robust Coreset for Continuous-and-Bounded Learning (with Outliers)

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this big data era, we often confront large-scale data in many machine learning tasks. A common approach for dealing with large-scale data is to build a small summary, *e.g.,* coreset, that can efficiently represent the original input. However, real-world datasets usually contain outliers and most existing coreset construction methods are not resilient against outliers (in particular, the outliers can be located arbitrarily in the space by an adversarial attacker). In this paper, we propose a novel robust coreset method for the *continuous-and-bounded learning* problem (with outliers) which includes a broad range of popular optimization objectives in machine learning, like logistic regression and k-means clustering. Moreover, our robust coreset can be efficiently maintained in fully-dynamic environment. To the best of our knowledge, this is the first robust and fully-dynamic coreset construction method for these optimization problems. We also conduct the experiments to evaluate the effectiveness of our robust coreset in practice.

1 Introduction

As the rapid increasing of data volume in this big data era, we often need to develop low-complexity (*e.g.,* linear or even sublinear) algorithms for machine learning tasks. Moreover, our dataset is often maintained in a dynamic environment so that we have to consider some more complicated issues like data insertion and deletion. For example, as mentioned in the recent article [12], Ginart *et al.* discussed the scenario that some sensitive training data has to be deleted due to the reason of privacy preserving. Obviously, it is prohibitive to re-train our model when the training data is changed dynamically, if the data size is extremely large. To remedy these issues, a natural way is to construct a small-sized summary of the training data so that we can run existing algorithm on the summary rather than the whole data. **Coreset** [10], which was originally studied in the community of computational geometry [1], has become a widely used data summary for large-scale machine learning. As a succinct data compression technique, coreset also enjoys a number of other nice properties. For instance, coreset is usually composable thus can be applied in the environment like distributed computing [17]. Also, small coreset can be obtained for the streaming algorithms [1, 4] and the fully-dynamic algorithms in the dynamic setting with data insertion and deletion [15].

However, the existing coreset construction methods are still far from being satisfactory in practice. A major bottleneck is that most of them are sensitive to outliers. We are aware that real-world dataset is usually noise and may contain outliers; note that the outliers can be located arbitrarily in the space and even one outlier can significantly destroy the final machine learning result. A typical example is poisoning attack, where an adversarial attacker may inject several specially crafted samples into the training data which can make the decision boundary severely deviate and cause unexpected misclassification [5]. To see why the existing coreset methods are sensitive to outliers, we can take the popular sampling based coreset framework [11] as an example. The framework needs

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

to compute a "sensitivity" for each data item, which measures the importance degree of the data item to the whole data set; however, it tends to assign high sensitivities to the points who are far from the majority of the data, that is, an outlier is likely to have a high sensitivity and thus has a high chance to be selected to the coreset. Obviously, the coreset obtained by this way is not appropriate since we expect to contain more inliers rather than outliers in the coreset. It is also more challenging to further construct a fully-dynamic robust coreset. The existing robust coreset construction methods [11, 16] often rely on simple uniform sampling and are efficient only when the number of outliers is a constant factor of the input size (we will discuss this issue in Section 3.1). Note that other outlier-resistant data summary methods like [13, 8] usually yield large approximation factors and are unlikely to be maintained in a fully dynamic scenario, to our knowledge.

1.1 Our Contributions

In this paper, we propose a **unified fully-dynamic robust coreset framework** for a class of optimization problems which is termed *continuous-and-bounded learning*. This type of learning problems actually covers a broad range of optimization objectives in machine learning [24, Chapter 12.2.2]. Roughly speaking, "continuous-and bounded learning" requires that the optimization objective is a continuous function (*e.g.,* smooth or Lipschitz), and meanwhile the solution is restricted within a bounded region. We emphasize that this "bounded" assumption actually is quite natural in real machine learning scenarios. We can consider the unsupervised optimization problem, *facility location*, as an example. Suppose we are going to build a supermarket in a city. Though there are many available candidate locations in the city, we often restrict our choice within some specific districts due to some other practical factors. Moreover, it is also reasonable to bound the solution range in a dynamic environment because one single update (insertion or deletion) is not quite likely to dramatically change the solution.

Our coreset construction is a novel hybrid framework. First, we suppose that there exists an ordinary coreset construction method $\mathcal{A}$ for the given continuous-and bounded optimization objective (without considering outliers). We partition the input data into two parts: the "suspected" inliers and the "suspected" outliers, where the ratio of the sizes of these two parts is a carefully designed parameter λ. For the "suspected" inliers, we run the method $\mathcal{A}$ (as a black box); for the "suspected" outliers, we directly take a small sample uniformly at random; finally, we prove that these two parts together yield a robust coreset. Our framework can be also efficiently implemented under the merge-and-reduce framework for dynamical setting (even though the original merge-and-reduce framework is not designed for the case with outliers) [4, 14]. A cute feature of our framework is that we can easily tune the parameter λ for updating our coreset dynamically, if the fraction of outliers is changed in the dynamic environment.

The other contribution of this paper is that we propose two different coreset construction methods for continuous-and-bounded optimization objectives (*i.e.,* the aforementioned black box $\mathcal{A}$). The first method is based on the importance sampling framework [11], and the second one is based on a space partition idea. Our coreset sizes depend on the doubling dimension of the solution space rather than the VC (shattering) dimension (this property is particularly useful if the VC dimension is too high or not easy to obtain). Our methods can be applied for a broad range of widely studied optimization objectives, such as logistic regression [22], Bregman clustering [2] and truth discovery [19]. It is worth noting that although some coreset construction methods have been proposed for some of these continuous-and-bounded objectives (like [22, 25, 21]), they are all problem-dependent and we are the first, to the best of our knowledge, to study them from a unified perspective.

2 Preliminaries

Suppose $\mathcal{P}$ is the parameter space. In this paper, we consider the learning problem whose objective function is the weighted sum of the cost over the training data, *i.e.*,

$$f(\theta, X) := \sum_{x \in X} w(x) f(\theta, x), \tag{1}$$

where X is the input data and $w(x) \geq 0$ indicates weight of each data item x; $f(\theta, x)$ is the cost contributed by x with the parameter vector $\theta \in \mathcal{P}$. The goal is to find an appropriate θ so that objective function $f(\theta, X)$ is minimized. Assuming that there are $z \in \mathbb{Z}^+$ outliers in X, we then

define the objective function with z outliers:

$$f_z(\theta, X) := \min_{O \subset X, [\![O]\!]=z} f(\theta, X \backslash O).$$

We use (X, z) to denote a given instance X with z outliers. Actually, the above definition comes from the popular "trimming" method [23] that has been widely used for robust optimization problems. Several other notations used throughout this paper is shown in Table 1. We always assume $|X| = n$ if the data items are unit-weighted; otherwise, we assume the total weight $[\![X]\!] = n$.

Table 1: List of Notations

$\lvert Y \rvert$	the number of data items in a set Y
$[\![Y]\!]$	the total weight of the data items in Y
$\mathbb{f}(\theta, X) := \frac{f(\theta, X)}{[\![X]\!]}$	the average form of $f(\theta, X)$
$\mathbb{f}_z(\theta, X) := \frac{1}{\lvert X \rvert - z} f_z(\theta, X)$	the average form of $f_z(\theta, X)$

We further impose some reasonable assumption to the function $f(\theta, x)$. Let $\mathcal{D}$ be a metric space. A function $g : \mathcal{D} \to \mathbb{R}$ is *α-Lipschitz continuous* if for any $\theta_1, \theta_2 \in \mathcal{D}$, $|g(\theta_1) - g(\theta_2)| \leq \alpha \|\Delta\theta\|$, where $\Delta\theta = \theta_1 - \theta_2$ and $\|\cdot\|$ is some specified norm in $\mathcal{D}$. A differentiable (*resp.,* twice-differentiable) function $g : \mathbb{R}^d \to \mathbb{R}$ is *α-Lipschitz continuous gradient* (*resp., α-Lipschitz continuous Hessian*) if its gradient (*resp.,* Hessian matrix) ∇g (*resp.,* $\nabla^2 g$) is α-Lipschitz continuous. An α-Lipschitz continuous gradient function is also called α-smooth. For any $\theta_1, \theta_2 \in \mathcal{D}$, the α-Lipschitz continuous gradient and α-Lipschitz continuous Hessian also imply the bounds of the difference between $g(\theta_1)$ and $g(\theta_2)$, which are

$$|g(\theta_1) - g(\theta_2) - \langle \nabla g(\theta_2), \Delta\theta \rangle| \leq \frac{\alpha}{2} \|\Delta\theta\|^2, \tag{2}$$

$$|g(\theta_1) - g(\theta_2) - \langle \nabla g(\theta_2), \Delta\theta \rangle - \frac{1}{2} \langle \nabla^2 g(\theta_2) \Delta\theta, \Delta\theta \rangle| \leq \frac{\alpha}{6} \|\Delta\theta\|^3. \tag{3}$$

Below we show the formal definition of the continuous-bounded learning problem.

Definition 1 (Continuous-and-Bounded Learning [24])**.** *Let α, $l > 0$. A learning problem is called* Lipschitz-and-bounded *with the parameters α, ℓ if the following two conditions hold:*

1. *There exists a fixed parameter vector $\tilde{\theta}$ such that θ is always restricted within $\mathcal{B}$, the ball centered at $\tilde{\theta}$ with radius ℓ in the parameter space $\mathcal{P}$.*
2. *For each $x \in X$, the loss function $f(\cdot, x)$ is α-Lipschitz.*

Similarly, we can define the smooth-and-bounded *and* Lipschitz Hessian-and-bounded *learning problems. In general, we call all these three problems* "continuous-and-bounded learning"*.*

As mentioned in Section 1.1, many practical optimization objectives fall under the umbrella of the continuous-and-bound learning problems.

We also define the coreset for continuous-and-bounded learning problems below. We assume each $x \in X$ has unit weight (*i.e.,* $w(x) = 1$), and it is not hard to extend our method to weighted case.

Definition 2 (ε-coreset and (ε, ν)-coreset)**.** *Let $\varepsilon, \nu > 0$. Given a dataset X and the objective function $f(\theta, X)$, we say that a weighted set C is an* ε-coreset *of X if for any $\theta \in \mathcal{B}$, we have*

$$|\mathbb{f}(\theta, C) - \mathbb{f}(\theta, X)| \leq \varepsilon \mathbb{f}(\theta, X). \tag{4}$$

The set C is called an (ε, ν)-coreset *if for any $\theta \in \mathcal{B}$,*

$$|\mathbb{f}(\theta, C) - \mathbb{f}(\theta, X)| \leq \varepsilon(\mathbb{f}(\theta, X) + \nu). \tag{5}$$

Remark 1. *The purpose for proposing the (ε, ν)-coreset is that we will use it as a bridge to achieve an ε-coreset in our analysis. Actually it is not difficult to see the relation between ε-coreset and (ε, ν)-coreset. Suppose $\varepsilon' = \min\{\varepsilon\pi/2\nu, \varepsilon/2\}$, where $\pi = \inf_{\theta \in \mathcal{B}} \mathbb{f}(\theta, X)$. Then an (ε', ν)-coreset should be an ε-coreset as well, because $\varepsilon'(\mathbb{f}(\theta, X) + \nu)$ is no larger than $\varepsilon \mathbb{f}(\theta, X)$ for any $\theta \in \mathcal{B}$.*

Following Definition 2, we define the corresponding *robust coreset* which was introduced in [11, 16] before.

Definition 3 (robust coreset)**.** *Let $\varepsilon, \nu > 0$ and $\beta \in (0,1)$. Given the dataset X and the objective function $f(\theta, x)$, we say that a weighted dataset C is a (β, ε)-robust coreset of X if for any $\theta \in \mathcal{B}$, we have*

$$(1-\varepsilon)\frac{f_{(1+\beta)z}(\theta, X)}{|X| - z} \leq \frac{f_z(\theta, C)}{[\![C]\!] - z} \leq (1+\varepsilon)\frac{f_{(1-\beta)z}(\theta, X)}{|X| - z}. \tag{6}$$

*The set C is called an $(\beta, \varepsilon, \nu)$-*robust coreset *if for any $\theta \in \mathcal{B}$,*

$$(1-\varepsilon)\frac{f_{(1+\beta)z}(\theta, X)}{|X| - z} - \varepsilon\nu \leq \frac{f_z(\theta, C)}{[\![C]\!] - z} \leq (1+\varepsilon)\frac{f_{(1-\beta)z}(\theta, X)}{|X| - z} + \varepsilon\nu \tag{7}$$

For the case without outliers, it is not difficult to see that the optimal solution of the ε-coreset is a $(1+3\varepsilon)$-approximate solution of the full data. Specifically, let θ^*_C be the optimal solution of an ε-coreset and θ^*_X be the optimal solution of the original dataset X; then for $\varepsilon \in (0, 1/3)$, we have

$$f(\theta^*_C, X) \leq (1+3\varepsilon) f(\theta^*_X, X). \tag{8}$$

But when considering outliers, this result only holds for $(0, \varepsilon)$-robust coreset (*i.e.,* $\beta = 0$). If $\beta > 0$, we can derive a weaker result.

Lemma 2. *Given two parameters $0 < \varepsilon < 1/3$ and $0 \leq \beta < 1/2$, suppose C is a (β, ε)-robust coreset of X with $(1+2\beta)z$ outliers. Let θ^*_C be the optimal solution of $(C, (1+2\beta)z)$, θ^*_X be the optimal solution of (X, z) respectively. Then we have*

$$f_{(1+4\beta)z}(\theta^*_C, X) \leq (1+3\varepsilon) f_z(\theta^*_X, X) \tag{9}$$

Proof. For $\beta \in [0, 1/2)$ and $\varepsilon \in (0, 1/3)$, we have $1 + 4\beta \geq (1+2\beta)(1+\beta)$, $(1+2\beta)(1-\beta) \geq 1$ and $\frac{1+\varepsilon}{1-\varepsilon} < 1 + 3\varepsilon$. Thus we can obtain the following bound:

$$\begin{aligned} f_{(1+4\beta)z}(\theta^*_C, X) &\leq f_{(1+2\beta)(1+\beta)z}(\theta^*_C, X) \leq \frac{1}{1-\varepsilon} f_{(1+2\beta)z}(\theta^*_C, C) \\ &\leq \frac{1}{1-\varepsilon} f_{(1+2\beta)z}(\theta^*_X, C) \leq \frac{1+\varepsilon}{1-\varepsilon} f_{(1+2\beta)(1-\beta)z}(\theta^*_X, X) \\ &\leq (1+3\varepsilon) f_z(\theta^*_X, X) \end{aligned}$$

□

The rest of this paper is organized as follows. In Section 3, we introduce our robust coreset framework and show how to realize it in a fully-dynamic environment. In Section 4, we propose two different ordinary coreset (without outliers) construction methods for continuous-and-bound learning problems, which can be used as the black box in our robust coreset framework of Section 3.

3 Our Robust Coreset Framework

As the warm-up, we first consider the simple uniform sampling as the robust coreset in Section 3.1. Then, we introduce our major contribution, the hybrid framework for robust coreset construction and its fully-dynamic realization in Section 3.2 and 3.3 respectively.

3.1 Warm-Up: Uniform Sampling

As mentioned before, the existing robust coreset construction methods [11, 16] are based on uniform sampling. Note that their methods are only for the clustering problems (*e.g.,* k-means/median clustering). Thus a natural question is that whether the uniform sampling idea also works for the general continuous-and-bounded optimization problems studied in this paper. Below, we answer this question in the affirmative.

Definition 4 (f-induced range space)**.** *Suppose $\mathcal{X}$ is an arbitrary metric space. Given the cost function $f(\theta, x)$ as (1) over $\mathcal{X}$, we let*

$$\mathfrak{R} = \Big\{ \{x \in \mathcal{X} : f(\theta, x) \leq \ell\} \mid \forall \ell \geq 0, \forall \theta \in \mathcal{P} \Big\}, \tag{10}$$

then $(\mathcal{X}, \mathfrak{R})$ is called the f-induced range space.

The following "δ-sample" concept comes from the theory of VC dimension [20]. Given a range space $(\mathcal{X}, \mathfrak{R})$, let C and X be two finite subsets of $\mathcal{X}$. Suppose $\delta \in (0,1)$. We say C is a *δ-sample* of X if $C \subseteq X$ and $\left|\frac{|X\cap R|}{|X|} - \frac{|C\cap R|}{|C|}\right| \leq \delta$ for any $R \in \mathfrak{R}$. Denote by $\texttt{vcdim}$ the VC dimension of the range space of Definition 4, then we can achieve a δ-sample with probability $1-\eta$ by uniformly sampling $O(\frac{1}{\delta^2}(\texttt{vcdim} + \log\frac{1}{\eta}))$ points from X [20]. The value of $\texttt{vcdim}$ depends on the function "f". For example, if "f" is the loss function of logistic regression in $\mathbb{R}^d$, then $\texttt{vcdim}$ can be as large as $\Theta(d)$ [22]. The following theorem shows that a δ-sample can serve as a robust coreset when z is a constant factor of n.

Theorem 1. *Let X be an instance of the continuous-and-bounded learning problem (2). If C is a δ-sample of X in the f-induced range space. Then we have*

$$f_{z+\delta n}(\theta, X) \leq f_z(\theta, C) \leq f_{z-\delta n}(\theta, X) \tag{11}$$

for any $\theta \in \mathcal{P}$ and any $\delta \in (0, z/n]$. In particular, if $\delta = \beta z/n$, C is a $(\beta, 0)$-robust coreset of X.

Proof. Given two δ-approximate subsets A and B of $\mathcal{X}$, where the range space is induced by $f(\theta, x)$. Let $|A| = a, |B| = b$ and $O = ab$. We create a new weighted set A' by setting the weight of each element in A to be b; it can be simply thought as making b copies of each element in A. Similarly, we create B' that each point has weight a. Then we have $[\![A']\!] = [\![B']\!] = O$.

We arrange elements in Y in the order of $f(\theta, y)$ and use y_i to denote the i-th element. First we claim that $f(\theta, a'_{i+\delta O}) \geq f(\theta, b'_i)$ for all $1 \leq i \leq (1-\delta)O$. If not, there must be some j with $f(\theta, a'_{j+\delta O}) < f(\theta, b'_j)$. Consider a set $R = \{x \in \mathcal{X} \mid f(\theta, x) \leq f(\theta, a'_{j+\delta O})\}$. We have $R \in \mathfrak{R}$, $[\![A' \cap R]\!] \geq j + \delta O$ and $[\![B' \cap R]\!] < j$. (We define the intersection of a weighted set and a unit-weighted set to be a weighted set such that each element should belong to both sets and it keeps its weight from the weighted set.) Then we have

$$\begin{aligned}
\frac{|A\cap R|}{|A|} &= \frac{[\![A'\cap R]\!]/b}{|A|} \geq \frac{(j+\delta O)/b}{a} = \frac{j}{O} + \delta \\
\frac{|B\cap R|}{|B|} &= \frac{[\![B'\cap R]\!]/a}{|B|} < \frac{j/a}{b} = \frac{j}{O},
\end{aligned}$$

A and B are not δ-approximate, a contradiction. Thus we prove the claim.
We denote by $\gamma = (n-z)/n$ the proportion of inliers, then we have

$$\begin{aligned}
f_{(1-\gamma)|A|}(\theta, A) &= \sum_{i=1}^{\gamma a} f(\theta, a_i) = \frac{1}{b}\sum_{i=1}^{\gamma O} f(\theta, a'_i) \\
&\geq \frac{1}{b}\sum_{i=1+\delta O}^{\gamma O} f(\theta, a'_i) \quad \text{because } f(\theta, \cdot) \geq 0 \\
&= \frac{1}{b}\sum_{i=1}^{(\gamma-\delta)O} f(\theta, a'_{i+\delta O}) \geq \frac{1}{b}\sum_{i=1}^{(\gamma-\delta)O} f(\theta, b'_i) \quad \text{by the claim } f(\theta, a'_{i+\delta O}) \geq f(\theta, b'_i) \\
&= \frac{a}{b}\sum_{i=1}^{(\gamma-\delta)b} f(\theta, y_i) \\
&= f_{(1-(\gamma-\delta))|B|}(\theta, B).
\end{aligned}$$

For simplicity we assume that δO is an integer, and it is easy to see that this result still holds when δO is not integral.

Let $A \leftarrow C$ and $B \leftarrow X$, we have $f_{z+\delta n}(\theta, X) \leq f_z(\theta, C)$. Let $B \leftarrow C, A \leftarrow X$ and $\gamma \leftarrow \gamma + \delta$, we have $f_z(\theta, C) \leq f_{z-\delta n}(\theta, X)$. □

Remark 3. *Though the uniform sampling is simple and easy to implement, it has two major drawbacks. First, it always involves an error on the number of outliers (otherwise, if $\delta = 0$, the sample should be the whole X). Also, it is useful only when z is a constant factor of n. For example, if $z = \sqrt{n}$, the obtained sample size can be as large as n. Our following hybrid robust framework in Section 3.2 can well avoid these two issues.*

3.2 The Hybrid Framework for (β, ε)-Robust Coreset

Our idea for building the robust coreset is inspired by the following intuition. In an ideal scenario, if we know who are the inliers and who are the outliers, we can simply construct the coresets for them separately. In reality, though we cannot obtain such a clear classification, the properties of the continuous-and-bounded objective function (from Definition 1) can guide us to obtain a "coarse" classification. Furthermore, together with some novel insights in geometry, we prove that such a hybrid framework can yield a (β, ε)-robust coreset.

Suppose the cost function f is continuous-and-bounded as Definition 1. Specifically, the parameter vector θ is always restricted within the ball centered at $\tilde{\theta}$ with radius ℓ. First, we partition X into two parts according to the value of $f(\tilde{\theta}, x)$. Let $Z := (1 + 1/\varepsilon)\, z$, and $x_c \in X$ be the point who has the Z-th largest cost $f(\tilde{\theta}, x)$ among X. We let $\tau = f(\tilde{\theta}, x_c)$, and thus we have

$$\left|\{x \in X : f(\tilde{\theta}, x) \geq \tau\}\right| = Z. \tag{12}$$

We call these Z points as the "suspected outliers" and the remaining $n - Z$ points as the "suspected inliers". If we fix $\theta = \tilde{\theta}$, the set of the "suspected outliers" contains at least $\frac{1}{\varepsilon} z$ real inliers (since $Z = (1 + 1/\varepsilon)\, z$). This immediately implies the following inequality:

$$\tau z \leq \varepsilon f_z(\tilde{\theta}, X). \tag{13}$$

Our robust coreset construction is as follows. Suppose we have an ordinary coreset construction method $\mathcal{A}$ as the black box (we will discuss it in Section 4). We build an ε-coreset for the suspected inliers by $\mathcal{A}$, and build a δ-sample for the suspected outliers with $\delta = \frac{\beta\varepsilon}{1+\varepsilon}$ (as Theorem 1). If we set $\beta = 0$, we just directly take all the suspected outliers as the δ-sample. We denote these two coresets as C_{si} and C_{so} respectively. Finally, we return $C = C_{si} \cup C_{so}$ as the robust coreset.

To prove the correctness of this construction, we imagine the following partition on X. For any parameter vector θ in the parameter space, X is also partitioned into the real inliers (*i.e.,* the set of $n - z$ points who have the smallest cost $f(\theta, x)$) and the real outliers (*i.e.,* the set of z points who have the largest cost $f(\theta, x)$). Therefore, together with the suspected inliers and outliers, X is partitioned into 4 parts:

- X_{I}: the points belonging to the real inliers and also the suspected inliers. X_{I} is not empty because we assume that suspected outliers are less than inliers ($Z < n - z$).
- X_{II}: the points belonging to the real inliers and the suspected outliers. X_{II} is not empty because the number of the suspected outliers is larger than the number of outliers ($Z > z$).
- X_{III}: the points belonging to the real outliers and also the suspected outliers. X_{III} can be empty iff all the real outliers are suspected inliers.
- X_{IV}: the points belonging to the real outliers and the suspected inliers. X_{IV} can be empty iff all the real outliers are suspected outliers.

Similarly, C is partitioned into 4 parts in the same way.

For continuous-and-bounded learning problems, we can bound $|f(\theta, x) - f(\tilde{\theta}, x)|$ by a low degree polynomial function. For example, if $f(\cdot, x)$ is α-Lipschitz, the polynomial is $\xi(\ell) = \alpha\ell$; if $f(\cdot, x)$ is α-smooth, the polynomial is $\xi(h; \ell) = h\ell + \alpha\ell^2/2$, where $h = \max_{x \in X} \|\nabla f(\theta, x)_{|\tilde{\theta}}\|$. For conciseness, we use the $\xi(\ell)$ in the following statement.

Theorem 2. *For continuous-and-bounded learning problems defined in Definition 1, given an ε-coreset method with size $|C_{si}|$ for the "suspected inliers", we can construct a $(\beta, 5\varepsilon, 2\xi(\ell))$-robust coreset method of size $|C_{si}| + \min\left\{O\left(\frac{1}{\beta^2\varepsilon^2}(\mathtt{vcdim} + \log\frac{1}{\eta})\right), O\left(\frac{z}{\varepsilon}\right)\right\}$ with probability at least $1 - \eta$. In particular, when $\beta = 0$, our coreset has no error on the number of outliers and its size is $|C_{si}| + O\left(\frac{z}{\varepsilon}\right)$.*

Proof (sketch). In this proof we omit technical details and the entire proof is put in the supplement.

Our aim is to prove that $f_z(\theta, C) \approx f_z(\theta, X)$. We have $f_z(\theta, C) = f(\theta, C_{\mathrm{I}} + C_{\mathrm{II}}) = f(\theta, C_{\mathrm{I}}) + f(\theta, C_{\mathrm{II}})$. Hence we need to bound $f(\theta, C_{\mathrm{I}})$ and $f(\theta, C_{\mathrm{II}})$ respectively. The upper bound of $f(\theta, C_{\mathrm{I}})$

comes from the definition of ε-coreset. We have $f(\theta, C_{\mathrm{I}}) \leq f(\theta, C_{\mathrm{I}}+C_{\mathrm{IV}}) \leq (1+\varepsilon) f(\theta, X_{\mathrm{I}}+X_{\mathrm{IV}})$ since $C_{\mathrm{I}} + C_{\mathrm{IV}}$ is an ε-coreset of $X_{\mathrm{I}} + X_{\mathrm{IV}}$.

The upper bound of $f(\theta, C_{\mathrm{II}})$ is a little complex. We give two different upper bounds depending on C_{IV} is empty or not:

$$f(\theta, C_{\mathrm{II}}) \leq \begin{cases} f_{(1-\beta)z}(\theta, X_{\mathrm{II}} + X_{\mathrm{III}}) & \text{if } C_{\mathrm{IV}} = \varnothing \\ f(\theta, X_{\mathrm{II}}) + 2z(\tau + \xi(\ell)) & \text{if } C_{\mathrm{IV}} \neq \varnothing \end{cases} \tag{14}$$

By adding the upper bound of $f(\theta, C_{\mathrm{I}})$ and $f(\theta, C_{\mathrm{II}})$, we obtain an upper bound of $f_z(\theta, C)$:

$$f_z(\theta, C) \leq \begin{cases} (1+\varepsilon) f_{(1-\beta)z}(\theta, X) + 4z\xi(\ell) & \text{if } C_{\mathrm{IV}} = \varnothing \\ (1+\varepsilon) f_z(\theta, X) + 4z\tau + 4z\xi(\ell) & \text{if } C_{\mathrm{IV}} \neq \varnothing \end{cases} \tag{15}$$

Merging two cases together, we have that

$$f_z(\theta, C) \leq (1+\varepsilon) f_{(1-\beta)z}(\theta, X) + 4z\tau + 4z\xi(\ell) \tag{16}$$

We have $z\tau \leq \varepsilon f_z(\theta, X) + \varepsilon(n-z)\xi(\ell)$ and $z\xi(\ell) \leq \varepsilon(n-z)\xi(\ell)$. Substitute these into (16) and take the average, we have

$$\frac{f_z(\theta, C)}{n-z} \leq (1+5\varepsilon) \frac{f_{(1-\beta)z}(\theta, X)}{n-z} + 10\varepsilon \cdot \xi(\ell) \tag{17}$$

Similarly, we can derive

$$\frac{f_z(\theta, C)}{n-z} \geq (1-5\varepsilon) \frac{f_{(1+\beta)z}(\theta, X)}{n-z} - 10\varepsilon \cdot \xi(\ell) \tag{18}$$

We conclude that C is a $(\beta, 5\varepsilon, 2\xi(\ell))$-robust coreset of X.

□

Finally, we can obtain a (β, ε)-robust coreset with the procedure stated in Remark 1.

3.3 The Fully-Dynamic Implementation

In this section, we show that our robust coreset of Section 3.2 can be efficiently implemented in a fully-dynamic environment, even if the number of outliers z is dynamically changed.

The standard ε-coreset usually has two important properties. If C_1 and C_2 are respectively the ε-coresets of two disjoint sets X_1 and X_2, their union $C_1 \bigcup C_2$ should be an ε-coreset of $X_1 \bigcup X_2$. Also, if C_1 is an ε_1-coreset of C_2 and C_2 is an ε_2-coreset of C_3, C_1 should be an $(\varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2)$-coreset of C_3. Based on these two properties, one can build a coreset for incremental data stream by using the merge-and-reduce technique [4, 14]. Very recently, Henzinger and Kale [15] extended it to the more general fully-dynamic setting, where data items can be deleted and updated as well.

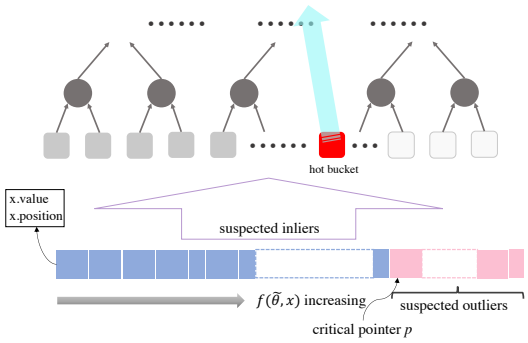

Figure 1: The illustration for our fully-dynamic robust coreset construction.

Roughly speaking, the merge-and-reduce technique uses a sequence of "buckets" to maintain the coreset for the input streaming data, and the buckets are merged by a bottom-up manner. However, it is challenging to directly adapt this strategy to the case with outliers, because we cannot determine the number of outliers in each bucket. A cute aspect of our hybrid robust coreset framework is that we can easily resolve this obstacle by using an $O(n)$ size auxiliary table $\mathscr{L}$ together with the merge-and-reduce technique (note that even for the case without outliers, maintaining a fully-dynamic coreset needs $\Omega(n)$ space [15]). Due to the space limit, we briefly introduce our idea below, and leave the full details to our supplement.

Recall that we partition the input data X into two parts: the $n - Z$ "suspected inliers" and the Z "suspected outliers", where $Z = (1 + 1/\varepsilon)z$. We follow the same notations used in Section 3.2. For the first part, we just apply the vanilla merge-and-reduce technique to obtain a fully-dynamic coreset C_{si}; for the other part, we can just take a δ-sample or take the whole set (if we require β to be 0), and denote it as C_{so}. Moreover, we maintain a table $\mathscr{L}$ to record the key values $x.value = f(\tilde{\theta}, x)$ and its position $x.position$ in the merge-and-reduce tree, for each $x \in X$; they are sorted by the $x.value$s in the table. To deal with the dynamical updates (*e.g.,* deletion and insertion), we also maintain a critical pointer p pointing to the data item x_c (recall x_c has the Z-th largest cost $f(\tilde{\theta}, x)$ among X defined in Section 3.2).

When a new data item x is coming or an existing data item x is going to be deleted, we just need to compare it with $f(\tilde{\theta}, x_c)$ so as to decide to update C_{si} or C_{so} accordingly; after the update, we also need to update x_c and the pointer p in $\mathscr{L}$. If the number of outliers z is changed, we just need to update x_c and p first, and then update C_{si} and C_{so} (for example, if z is increased, we just need to delete some items from C_{so} and insert some items to C_{si}). To assist these updating operations, we also set one bucket as the "hot bucket", which serves as a shuttle to execute all the data shifts. See Figure 1 for the illustration. Let M be the size of each leaf bucket; our time complexity for insertion and deletion is $O(M \log n)$; for updating z to $z \pm \Delta z$, our time complexity is $O(\frac{\Delta z}{\varepsilon} M \log n)$, where ε is the error bound for the robust coreset in Definition 3.

4 Coreset for Continuous-Bounded Learning Problems

As mentioned in Section 3.2, we need a black box ordinary coreset (without considering outliers) construction method $\mathcal{A}$ in the hybrid robust coreset framework. In this section, we provide two general ε-coreset construction methods for the continuous-and-bounded learning problems.

4.1 Importance Sampling Based Coreset Construction

We follow the importance sampling based approach [18]. For each data point x, it has a sensitivity $\sigma(x) = \sup_\theta \frac{f(\theta,x)}{f(\theta,X)}$ that measures its importance to the whole input data X. Computing the sensitivity is often challenging but an upper bound of the sensitivity actually is often sufficient for the coreset construction. Suppose $X = \{x_1, \cdots, x_n\}$ and we denote $\sigma(x_i)$ by σ_i for convenience. Assume s_i is an upper bound of σ_i and let $S = \sum_{i=1}^n s_i$. The coreset construction is as follows. We sample a subset C from X, where each element of C is sampled *i.i.d.* with probability $p_i = s_i/S$; we assign a weight $w_i = \frac{S}{s_i|C|}$ to each sampled data item x_i of C.

Theorem 3 ([6]). *Let* `vcdim` *be the VC dimension (or shattering dimension) of the range space induced from $f(\theta, x)$. If the size of C is $\Theta\left(\frac{S}{\varepsilon^2}\left(\texttt{vcdim} \cdot \log S + \log \frac{1}{\eta}\right)\right)$, then C is an ε-coreset with probability $1 - \eta$.*

Therefore the only remaining issue is how to compute the s_is. We denote $f(\theta, x_i)$ by $f_i(\theta)$ for short. Recall that we assume our cost function to be α-Lipschitz (or α-smooth, α-Lipschitz continuous Hessian) in Definition 1. That is, we can bound the difference between $f_i(\theta)$ and $f_i(\tilde{\theta})$. For example, if we assume the cost function to be α-smooth, we have $f_i(\theta) \leq f_i(\tilde{\theta}) + \langle \nabla f_i(\tilde{\theta}), \Delta\theta \rangle + \frac{\alpha}{2}\|\Delta\theta\|^2$ and $f_i(\theta) \geq f_i(\tilde{\theta}) + \langle \nabla f_i(\tilde{\theta}), \Delta\theta \rangle - \frac{\alpha}{2}\|\Delta\theta\|^2$, where $\Delta\theta = \theta - \tilde{\theta}$. Consequently, we obtain an upper bound of σ_i:

$$\sup_{\Delta\theta \in \mathbb{R}^d, \|\Delta\theta\| \leq \ell} \frac{f_i(\tilde{\theta}) + \langle \nabla f_i(\tilde{\theta}), \Delta\theta \rangle + \frac{\alpha}{2}\|\Delta\theta\|^2}{\sum_{i=1}^n f_i(\tilde{\theta}) + \langle \nabla f_i(\tilde{\theta}), \Delta\theta \rangle - \frac{\alpha}{2}\|\Delta\theta\|^2}. \tag{19}$$

In our supplement, we further show that computing such an upper bound is equivalent to solving a quadratic fractional programming. This programming can be reduced to a semi-definite programming [3], which can be solved in $O(\texttt{poly}(d))$ time. So the total running time of the coreset construction is $O(n \cdot \texttt{poly}(d))$.

A drawback of Theorem 3 is that the coreset size depends on the VC dimension `vcdim` induced by $f(\theta, x)$. For some objectives, `vcdim` can be very large or difficult to obtain. Here, we prove that for a continuous-and-bounded cost function, the coreset size can be independent of `vcdim`; instead,

it depends on the doubling dimension $\texttt{ddim}$ [7] of the parameter space $\mathcal{P}$. Doubling dimension is a widely used measure to describe the growth rate of the data, which can also be viewed as a generalization of the Euclidean dimension. For example, the doubling dimension of a d-dimensional Euclidean space is $\Theta(d)$. The proof of Theorem 4 is placed in the supplement.

Theorem 4. *Suppose the cost function $f(\theta, x)$ is continuous-and-bounded as described in Definition 1, and let* $\texttt{ddim}$ *be the doubling dimension of the parameter space. Then, if we run the importance sampling based coreset construction method with the sample size $|C| = \Theta\left(\frac{S^2}{\varepsilon^2}\left(\texttt{ddim}\log\frac{1}{\varepsilon} + \log\frac{1}{\eta}\right)\right)$, C is an ε-coreset with probability $1-\eta$. The hidden constant of $|C|$ depends on α and $\inf_{\theta\in\mathcal{B}} \mathbb{f}(\theta, X)$.*

Remark 4. *The major advantage of Theorem 4 over Theorem 3 is that we do not need to know the VC dimension induced by the cost function. On the other hand, the doubling dimension is often much easier to know (or estimate),* e.g., *the doubling dimension of a given instance in $\mathbb{R}^d$ is just $\Theta(d)$, no matter how complicated the cost function is. The reader is also referred to [16] for a more detailed discussion on the relation between VC (shattering) dimension and doubling dimension.*

4.2 Spatial Partition Based Coreset Construction

The reader may realize that the coreset size presented in Theorem 4 (and also Theorem 3) is **data-dependent**. That is, the coreset size depends on the value S, which can be different for different input instance X (because the formula (19) depends on X). To achieve a **data-independent** coreset size, we introduce the following method based on spatial partition, which is partly inspired by the previous k-median/means clustering coreset construction idea of [9]. We extend their method to the continuous-and-bounded learning problems and call is as *Generalized Exponential Layer (GEL)* method.

We set $\varrho = \min_{x\in X} f(\tilde{\theta}, x)$ and $T = \frac{1}{|X|} f(\tilde{\theta}, X)$. Then, we partition all the data points to different layers according to their cost with respect to $\tilde{\theta}$. Specifically, we assign a point x to the $\lfloor \log(\frac{f(\tilde{\theta},x)-\varrho}{T}) \rfloor$-th layer if $f(\tilde{\theta}, x) - \varrho \geq T$; otherwise we assign it to the 0-th layer. It is easy to see the number of the layers L is $O(\log n)$. We denote the set of points falling in the j-th layer as X_j. From each X_j, we take a small sample C_j uniformly at random, where each point of C_j is assigned the weight $|X_j|/|C_j|$. Finally, the union set $\bigcup_{j=0}^{L} C_j$ form our final coreset.

Theorem 5. *Suppose the cost function $f(\theta, x)$ is continuous-and-bounded as described in Definition 1, and let* $\texttt{ddim}$ *be the doubling dimension of the parameter space. The above coreset construction method GEL can achieve an ε-coreset of size $\Theta\left(\frac{\log n}{\varepsilon^2}\left(\texttt{ddim}\log\frac{1}{\varepsilon} + \log\frac{1}{\eta}\right)\right)$ in linear time. The hidden constant of $|C|$ depends on α and $\inf_{\theta\in\mathcal{B}} \mathbb{f}(\theta, X)$.*

To prove Theorem 5, the key is show that each C_j can well represent the layer X_j with respect to any θ in the bounded region (*i.e.,* the ball centered at $\tilde{\theta}$ with radius l in the parameter space as described in Definition 1). First, we use the continuity property to bound the difference between $f(\theta, x)$ and $f(\tilde{\theta}, x)$ for each $x \in X_j$ with a fixed θ; then, together with the doubling dimension, we can generalize this bound to any θ in the bounded region. The full proof is shown in the supplement.

5 Conclusion and Future Work

In this paper, we propose a robust coreset framework for the continuous-and-bounded learning problems (with outliers). Our framework can be efficiently implemented in the dynamical setting. We put our experimental results in the supplement due to the space limit. In future, it is interesting to consider constructing (dynamic) robust coresets for other types of optimization problems in machine learning.

References

[1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, 2004.

[2] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *J. Mach. Learn. Res.*, 6:1705–1749, 2005.

[3] A. Beck and M. Teboulle. A convex optimization approach for minimizing the ratio of indefinite quadratic functions over an ellipsoid. *Mathematical Programming*, 118:13–35, 2009.

[4] J. L. Bentley and J. B. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980.

[5] B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognit.*, 84:317–331, 2018.

[6] V. Braverman, D. Feldman, and H. Lang. New frameworks for offline and streaming coreset constructions. *CoRR*, abs/1612.00889, 2016.

[7] T. H. Chan, A. Gupta, B. M. Maggs, and S. Zhou. On hierarchical routing in doubling metrics. *ACM Trans. Algorithms*, 12(4):55:1–55:22, 2016.

[8] J. Chen, E. S. Azer, and Q. Zhang. A practical algorithm for distributed clustering and outlier detection. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 2253–2262, 2018.

[9] K. Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM J. Comput.*, 39(3):923–947, 2009.

[10] D. Feldman. Introduction to core-sets: an updated survey. *CoRR*, abs/2011.09384, 2020.

[11] D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In L. Fortnow and S. P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 569–578. ACM, 2011.

[12] A. Ginart, M. Y. Guan, G. Valiant, and J. Zou. Making AI forget you: Data deletion in machine learning. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3513–3526, 2019.

[13] S. Gupta, R. Kumar, K. Lu, B. Moseley, and S. Vassilvitskii. Local search methods for k-means with outliers. *Proc. VLDB Endow.*, 10(7):757–768, Mar. 2017.

[14] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In L. Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 291–300. ACM, 2004.

[15] M. Henzinger and S. Kale. Fully-dynamic coresets. In F. Grandoni, G. Herman, and P. Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 57:1–57:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[16] L. Huang, S. H. Jiang, J. Li, and X. Wu. Epsilon-coresets for clustering (with outliers) in doubling metrics. In M. Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 814–825. IEEE Computer Society, 2018.

[17] P. Indyk, S. Mahabadi, M. Mahdian, and V. S. Mirrokni. Composable core-sets for diversity and coverage maximization. In R. Hull and M. Grohe, editors, *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 100–108. ACM, 2014.

[18] M. Langberg and L. J. Schulman. Universal epsilon-approximators for integrals. In M. Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 598–607. SIAM, 2010.

[19] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han. A survey on truth discovery. *SIGKDD Explor.*, 17(2):1–16, 2015.

[20] Y. Li, P. M. Long, and A. Srinivasan. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences*, 62(3):516–527, 2001.

[21] M. Lucic, O. Bachem, and A. Krause. Strong coresets for hard and soft bregman clustering with applications to exponential family mixtures. In A. Gretton and C. C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, volume 51 of *JMLR Workshop and Conference Proceedings*, pages 1–9. JMLR.org, 2016.

[22] A. Munteanu, C. Schwiegelshohn, C. Sohler, and D. P. Woodruff. On coresets for logistic regression. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6562–6571, 2018.

[23] P. J. Rousseeuw and A. Leroy. *Robust Regression and Outlier Detection*. Wiley, 1987.

[24] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014.

[25] E. Tolochinsky and D. Feldman. Generic coreset for scalable learning of monotonic kernels: Logistic regression, sigmoid and more, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Some proofs are put in the supplement.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In the supplemental material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]