Data-Efficient Learning via Clustering-Based Sensitivity Sampling: Foundation Models and Beyond

Kyriakos Axiotis [* 1] **Vincent Cohen-Addad** [* 1] **Monika Henzinger** [* 2] **Sammy Jerome** [* 1] **Vahab Mirrokni** [* 1]
David Saulpic [* 3] **David P. Woodruff** [* 1 4] **Michael Wunder** [* 1]

Abstract

We study the data selection problem, whose aim is to select a small representative subset of data that can be used to efficiently train a machine learning model. We present a new data selection approach based on k-means clustering and sensitivity sampling. Assuming access to an embedding representation of the data with respect to which the model loss is Hölder continuous, our approach provably allows selecting a set of "typical" $k + 1/\varepsilon^2$ elements whose average loss corresponds to the average loss of the whole dataset, up to a multiplicative $(1 \pm \varepsilon)$ factor and an additive $\varepsilon\lambda\Phi_k$, where Φ_k represents the k-means cost for the input embeddings and λ is the Hölder constant. We furthermore demonstrate the performance and scalability of our approach on fine-tuning foundation models and show that it outperforms state-of-the-art methods. We also show how it can be applied on linear regression, leading to a new sampling strategy that surprisingly matches the performance of leverage score sampling empirically, while being conceptually simpler and more scalable.

1. Introduction

The growth of both datasets and models to a massive scale has led to a new generation of machine learning models with astonishing performance. Yet, the size of these models and datasets makes their training and fine-tuning extremely difficult, costly, time-consuming, and so nearly impossible for most academic institutions or small-scale companies to

perform. On the other hand, a complete dataset is often not needed to reach nearly optimal performance (i.e., up to a small increase in error percentage). A central question then becomes how to identify the most important data items for the training or fine-tuning process.

While uniform sampling often shows surprisingly good performance, it is still suboptimal, especially when dealing with real datasets that are complex and imbalanced. To better capture the usefulness of the underlying data to train the model, data selection and active learning methods deduce which data items are the most relevant for training or fine-tuning, based on their uniqueness, quality, and the model's knowledge. There exist several heuristics or greedy approaches for active learning and data selection (see e.g. (Dasgupta, 2004) or references in (Ren et al., 2021)). State-of-the-art data selection strategies are uncertainty-based, e.g., *margin* or *entropy* scores, and aim at selecting items for which the uncertainty of the model is high (e.g., (Balcan et al., 2007)). However, such purely model-based methods have the additional overhead of requiring evaluating the model on the whole dataset.

In a celebrated result, Sener and Savarese (Sener & Savarese, 2018) showed that state-of-the-art active learning strategies are difficult to use in modern training frameworks for the following reasons:

1. The training proceeds in batches, which requires the active learning strategy to pick *not only one* training element at a time but a *batch* of training elements. However, to make the most out of the batch, it is needed to ensure some diversity in the set of elements sampled, which often unfortunately anti-correlates with, for example, the margin objective which may lead to picking near-duplicate elements (see the discussion in (Sener & Savarese, 2018) for more details).

2. The score (i.e., value) of the training elements is obtained through the model. This requires running the model on the data items to determine which ones to pick next. Unfortunately, modern models are often very large and the inference time is particularly costly. Moreover, margin scores are not well-suited for foun-

---

*Equal contribution [1]Google Research [2]Institute for Science and Technology Austria (ISTA), Klosterneuburg, Austria [3]CNRS & IRIF, Université Paris Cité, Paris, France [4]Carnegie Mellon University. Correspondence to: Vincent Cohen-Addad <cohenaddad@google.com>, David Saulpic <david.saulpic@irif.fr>.

dation models.

The solution proposed by (Sener & Savarese, 2018) consists of using the notion of *coresets* to select the data. A coreset is a subset of the data defined such that optimizing the model on the coreset yields a good model for the entire dataset (i.e., good generalization bounds for the whole dataset). In more formal terms, the average model's gradient (or model loss) of the coreset elements is the same as the model's gradient of the whole dataset and thus, learning on the coreset elements has the same effect as learning on the whole dataset. Unfortunately, to implement this approach one would need to obtain the gradient or loss of *all* the data items, which implies running the model on all the data items. To circumvent this problem, Sener and Savarese show that, given some embedding representation of the dataset and some set of assumptions relating the embeddings to the model's loss, some coreset can be computed using a heuristic to the k-center objective. Their embedding assumption is a fairly natural one since the embeddings can be obtained from a pre-trained model or from a generic embedding model (e.g., BERT (Devlin et al., 2018), word2vec (Mikolov et al., 2013)). This approach has, however, the following suboptimal behavior:

(1) The first practical issue is that the k-center objective is particularly sensitive to outliers, and in particular the greedy 2-approximation algorithm in the work of (Sener & Savarese, 2018). Indeed the algorithm iteratively picks the training items that are the furthest away (in the embedding space) from the already selected training items. This tends to select outliers, increasing the diversity at the expense of the relevance of the elements. We ask: Can we find a more robust way of selecting a set of items which is both diverse and that precisely covers the most important traits of the dataset?

(2) A second theoretical drawback is that the bounds proven are quite weak and require strong assumptions on the relationship between the embeddings of the training elements and the model loss, in particular on the spread of the data elements (see Section 2.1 for more details). We ask: Can we provide a theoretical solution that would require a minimal set of assumptions on our dataset and model?

(3) Finally, and maybe most importantly, their approach is limited to classification tasks. We ask: Can we provide a more generic data-selection algorithm, working for a more general loss function and in particular for the new generation of foundation models?

1.1. Our Approach and Contribution

Consider the problem of fine-tuning a Large Language Model (LLM) on a specialized task, such as translation. Even though we have abundant data points for the translation task, it is often time-consuming and costly to fine-tune on the whole translation dataset, and we would instead prefer to sample a small representative subset of data that can still be used to build a high quality model. While there are methods that compute importance scores for each data point (e.g., margin scores) that can then be used to select data, these scores are expensive to compute, since they require evaluating *all* the data using the LLM.

Our key insight is to leverage such expensive, accurate scores on a *sublinear* number of data points, coupled with less accurate but fast to compute embeddings, which can be generated by a much simpler and efficient model. Surprisingly, we find that even simple embeddings, such as those generated by a pre-trained BERT model (Devlin et al., 2018; Mikolov et al., 2013), can be predictive of the loss affinity between different data points, for much larger and more complex models (see also Figure 1). Our data selection algorithm utilizes clustering and sketching techniques, offering strong theoretical guarantees and significant practical improvements over existing methods on benchmark datasets. The power of our theoretical contribution is that while the analysis is quite simple, it significantly improves over previous work, in particular over Sener & Savarese (2018).

Ideally, we want to sample data proportional to the model loss. However, like margin or entropy scores, this is expensive due to requiring model evaluation on the entire dataset. Instead, we leverage embeddings to identify a diverse and relevant subset. Slightly more formally, the data-selection problem that we consider is, similar to Sener and Savarese, the following. We assume that we are given a dataset $\mathcal{D}$, together with a loss function ℓ. The goal is to sample $S \subseteq \mathcal{D}$ of limited size, such that the total loss evaluated on the small sample S is the same as that evaluated on the full dataset $\mathcal{D}$. Or in other words that S is an unbiased estimator of the loss of $\mathcal{D}$. For scalability purposes, the algorithm computing S should make a sublinear number of queries to the loss function. We formalize this definition in Definition 3.

Our approach begins with k-means clustering on the entire dataset. Then, elements are sampled using *sensitivity sampling* on *proxy losses*. Specifically, the algorithm first computes a k-means clustering on the whole dataset and then samples each element with probability proportional to its distance to the closest mean plus the mean's loss (see (Feldman & Langberg, 2011) for the introduction of that probability distribution for clustering coresets, see also (Bachem et al., 2018; Bansal et al., 2024)), or of coresets in the sublinear time regime (Cohen-Addad et al., 2021c).

Here, we use a k-means clustering (or, more generally, (k, z)-clustering objective, which is k-median for $z = 1$ and k-means for $z = 2$) because it provides a more robust clustering measure than k-center as it is much less sensitive to outliers.

In the experiments section 5.2, we show that this yields a better sample in practice than what was previously known, in particular for fine-tuning a foundation model (specifically for fine-tuning an LLM to a translation task). We further show that our method is quite general as it works for both neural networks and for regression tasks.

Next, we provide theoretical guarantees on this sampling strategy. First, assuming there is a clustering such that, for each cluster, the model loss is Hölder continuous with respect to the embeddings – a well-motivated assumption as we show in the experiments section and less restrictive than the Lipschitz assumption of Sener & Savarese (2018) – we can prove that the samples provide a strong proxy for the loss of all the data elements. More specifically, we show that by using sensitivity sampling we obtain a coreset whose average model loss is within a $(1 \pm \varepsilon)$ factor of the average model loss on the whole dataset, plus an additive term corresponding to the loss of the (k, z)-clustering objective. This implies that if the embeddings of the data are clusterable (i.e., they have small (k, z)-clustering loss), we obtain an actual coreset for the model loss with only few *inferences*, i.e., queries to the loss function ℓ. Moreover, for classification tasks, expecting that the model embeddings will be clusterable is not an unrealistic assumption: we do expect that points from the same class have closer model-embedding distance than points in different classes. This intuition is validated in Table 1.

More formally, we work with the notion of Hölder continuity: we say that the loss function ℓ is (z, λ)-Hölder continuous if for any x, y with embeddings $E(x)$ and $E(y)$, $|\ell(x) - \ell(y)| \leq \lambda \|E(x) - E(y)\|^z$. We make the following assumption on the loss function:

Assumption 1. *For $\Lambda = (\Lambda_1, \ldots, \Lambda_k) \in \mathbb{R}^k$, an embedding E and a k-clustering of the input $\mathcal{C}$, we say the loss function is (z, Λ)-well-behaved with respect to E and $\mathcal{C}$ when, for any cluster C_i and point $e \in C_i$, $|\ell(e) - \ell(c_i)| \leq \Lambda_i \|E(e) - E(c_i)\|^z$.*

Note that this definition generalizes Lipschitzness: if the loss function is λ-Lipschitz, then the above holds for $\Lambda_i = \lambda$ and $z = 1$, regardless of the clustering $\mathcal{C}$. For simplicity, we will often write e for the embedding $E(e)$.

In Table 1, we validate Assumption 1 on standard benchmark datasets. We cluster the data, and then compute the percentiles of the ratio $|\ell(e) - \ell(c_i)| / \|e - c_i\|^z$ (for MNIST ℓ is the loss function, while for GAS it is the target variable). We observe that these values are indeed small, implying that the MNIST and GAS datasets do possess the (z, λ)-Hölder condition for a small constant λ.

We also sanity check our assumption on LLMs. Specifically, we examine how predictive a BERT-based embedding clustering is of the losses on a much larger T5 transformer

Table 1. Value of the Hölder continuous constant for $z = 2$ for the different percentiles of the MNIST (classification) and GAS (regression) datasets.

Smallest %ile	λ for MNIST	λ for GAS
20	0.00111	0.02286
40	0.00320	0.04488
60	0.00663	0.07158
80	0.01416	0.12589
100	0.05935	0.86976

model. Figure 1 shows that on average a data point's loss on the T5 model is much closer to the loss of similarly clustered points than that of random points.[1]
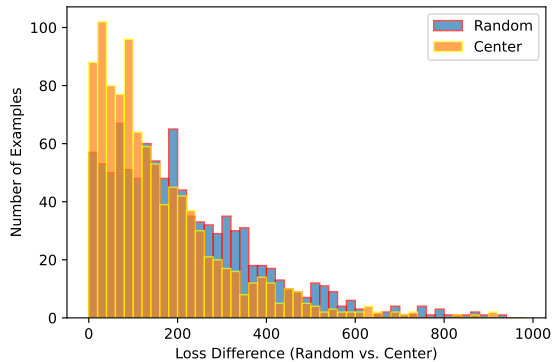

Figure 1. Distribution of loss to random point vs. center of the corresponding cluster for the WMT T2T EnDe translation dataset (Bojar et al., 2014) using BERT embeddings (Devlin et al., 2018).

We now proceed to state our main theorem, which gives strong data selection guarantees under Assumption 1.

Theorem 2. *Let $\varepsilon, z > 0$, $\Lambda \in \mathbb{R}^k$. Let $\mathcal{D}$ be a dataset and ℓ a loss function that is (z, Λ)-well-behaved with respect to an embedding E and a clustering $(C_1, \ldots, C_k)$ into k clusters. Then, there exists an algorithm that makes k queries to ℓ and outputs a sample S of size $O(\varepsilon^{-2})$ and a weight function w such that*

$$|\sum_{e \in \mathcal{D}} \ell(e) - \sum_{e \in S} w(e)\ell(e)| \leq \varepsilon \left(\sum_{e \in \mathcal{D}} \ell(e) + 2\Phi_{\mathcal{C},z}^{\Lambda}(\mathcal{D}) \right),$$

with constant probability, where $\Phi_{\mathcal{C},z}^{\Lambda}(\mathcal{D})$ =

---

[1]In numbers, the average absolute loss difference between two random points is 242.86; whereas the average absolute loss difference between a point and its center is 175.20. We ran a similar analysis as well using only random points - we selected 2 random points, and computed the average loss if the distance between them was "small" (< 5 in the embedding space) vs. the average loss when the distance was "large" (> 5 in the embedding space). The "large" average was 253.78, whereas the "small" average 215.66. We note we ran the analysis 100 times to ensure statistical significance.

$\sum_{i=1}^{k} \Lambda_i \Phi_{1,z}(C_i)$, *namely the (k,z)-clustering cost where cluster i is weighted by Λ_i.*

To interpret this result better note that when the loss function is (λ, z)-Hölder continuous, the upper-bound becomes $\varepsilon(\sum_{e \in \mathcal{D}} \ell(e) + 2\lambda \Phi_{k,1}(\mathcal{D}))$, where $\Phi_{k,z}(\mathcal{D})$ is the (k,z)-clustering cost of the dataset (which is k-means when $z=2$, or k-median when $z=1$ and the loss is λ-Lipschitz). We believe that our condition is quite a weak requirement; furthermore, we show that such a condition is needed: *There exist worst-case loss functions that are not Hölder continuous and for which it is necessary to query the whole dataset in order to get the above theorem statement* (see Theorem 5). Thus, this answers the second question raised above.

The assumption on the clustering can be read as follows: we expect the embedding to have a clustered structure that an algorithm (e.g., k-means++) can discover. *We expect elements in each cluster to be similar, and therefore that the loss function is smooth within each cluster* – this is formalized by the Hölder continuity. Note that the clustering may not be optimal, we only need to be able to compute it efficiently.

The additive error in our results depends on the (k,z)-clustering cost, which is defined as $\min_{|C|=k} \sum_{e \in \mathcal{D}} \min_{c \in C} \|e - c\|^z$. Therefore, depending on the choice of z, our upper bound can be made more, or less, robust to outliers: the smaller the z the more resilient to outliers it becomes; for sufficiently large z the objective becomes the k-center objective of (Sener & Savarese, 2018), which would translate here (with the assumption that the loss is λ-Lipschitz) into an upper bound $n \cdot \lambda \cdot \min_{|C|=k} \max_{e \in \mathcal{D}} \min_{c \in C} \|e - c\|$. This addresses question 1 raised above.

We further demonstrate that the resulting sampling strategy outperforms classic data selection approaches, namely, training the model using the set S obtained via Theorem 2 gives a several percentage increase in accuracy (more than 4% for Fashion MNIST) than using other methods. Similarly, for linear regression, we show empirically that our sampling strategy is competitive and sometimes outperforms more sophisticated state-of-the-art methods, such as leverage score sampling, adding a fundamentally new sampling strategy to the growing body of work on active regression (Chen & Price, 2019; Chen & Derezinski, 2021; Parulekar et al., 2021; Musco et al., 2022; Woodruff & Yasuda, 2023).

We defer a detailed survey of related work to Appendix A.1

2. Problem Formulation

Given a dataset $\mathcal{D}$ and a machine learning model, the high-level goal is to find a subset S of $\mathcal{D}$ such that training the model on S yields approximately the same model as training

the model on $\mathcal{D}$, while the time taken to compute S and train the model on S should be much smaller than the time taken to train the model on $\mathcal{D}$.

We focus here on the general data selection problem, and dedicate Section 4 to the special case of linear regression.

2.1. Our Model

We assume that we are given a dataset $\mathcal{D}$ of size n, together with a loss function ℓ such that $\ell(e)$ is the loss of the model on instance e. The goal is to sample $S \subseteq \mathcal{D}$ of limited size, and associate a weight function $w : S \mapsto \mathbb{R}_+$ such that

$$\Delta(S) := |\sum_{e \in \mathcal{D}} \ell(e) - \sum_{e \in S} w(e)\ell(e)| \leq \delta,$$

for the smallest possible δ. Note that $\ell(e)$ can be queried simply by running the model on e and computing the loss for e, which is expensive. This is why we want to compute S without having to compute $\ell(e)$ for all $e \in \mathcal{D}$. We now provide a complete formulation of the problem.

Definition 3 (Data Selection, (Sener & Savarese, 2018))**.** *The data selection problem is defined as follows:*

- ***Input:*** *A dataset $\mathcal{D}$, oracle access to a function $\ell : \mathcal{D} \mapsto \mathbb{R}_+$, and a target size s.*

- ***Output:*** *A sample $S \subseteq \mathcal{D}$ of size at most s together with a weight function $w : S \mapsto \mathbb{R}_+$ such that*

  – *The number of queries to ℓ (i.e.: inferences) is at most s.*
  – *S minimizes*

$$\Delta(S) := |\sum_{e \in \mathcal{D}} \ell(e) - \sum_{e \in S} w(e)\ell(e)|. \quad (1)$$

Note that our definition introduces two differences from the the original definition of (Sener & Savarese, 2018). (A) First, they use uniform weights, namely $\forall s \in S, w(s) = \frac{|\mathcal{D}|}{|S|}$ (in which case, minimizing Equation (1) means that the average $\ell(e)$ in the sample should be close to the average $\ell(e)$ for the whole data). We slightly generalize the definition to allow for different sampling strategies, while keeping an unbiased estimator.

(B) Second, (Sener & Savarese, 2018) consider the loss after re-training the model with S, namely $\left|\frac{1}{n}\sum_{e \in \mathcal{D}} \ell(e, \mathcal{A}(S)) - \sum_{e \in S} w(e)\ell(e, \mathcal{A}(S))\right|$. In those notations, $\mathcal{A}(S)$ represents the model parameters after training on S, and the loss function ℓ has two parameters to emphasize its dependency in the model's parameters. In words, the loss of the model trained on S is roughly the same evaluated on S as on $\mathcal{D}$. In order to bound this quantity, Sener and Savarese make *strong assumptions on the*

distribution of the dataset, namely the labels are drawn randomly from a structured distribution, and it is further assumed that the training loss is 0 on their sample. Instead, we stay more general and focus on the loss in the current model. Our underlying assumption is that, if S approximates the loss well, then it contains "typical" items of the dataset (with respect to the current model), and therefore training the model based on S should be similar as training it on $\mathcal{D}$. This formulation allows us to show strong theoretical results for the data selection problem, *without any assumptions on $\mathcal{D}$*. Note that our objective is more challenging than the one from Sener and Savarese: the bound we prove on $\Delta(S)$ implies the result of (Sener & Savarese, 2018) (under their assumption about the model loss).

2.2. Assumptions on ℓ

Limits to the general case   The above formulation requires that the number of queries to ℓ is sublinear in $|\mathcal{D}|$. Unfortunately, as long as $s = o(|\mathcal{D}|)$ it is impossible to bound $\Delta(S)$ without further assumptions on ℓ or $\mathcal{D}$, which can be seen by the following worst-case instance: The adversary chooses uniformly at random (u.a.r.) an element $e^* \in \mathcal{D}$ and defines $\ell(e^*) = 1$ and $\ell(e) = 0$ for all $e \neq e^*$. Then computing with constant success probability a sample S of size $s = o(\mathcal{D})$ such that $\Delta(S) = o(\sum_{e \in \mathcal{D}} \ell(e))$ with $o(|\mathcal{D}|)$ queries to ℓ is impossible. Of course, this worst-case instance is unrealistic and we can hope to better capture the structure of real-world dataset.

Assumption on the embeddings   In practice we can assume that each element e in $\mathcal{D}$ could be associated with a vector $E(e)$ in $\mathbb{R}^d$ for some d, possibly coming from a model that is "well-behaved" with respect to the loss function of the model. More concretely, the embeddings of the data elements can either be obtained from a generic embedding of the input dataset $\mathcal{D}$, e.g., the BERT or word2vec embeddings for words (Devlin et al., 2018; Mikolov et al., 2013), or an embedding obtained through the last layers of the model being trained. The last assumption is particularly realistic in the *warm start* or *fine-tuning* regime where the model has already been partially trained.

Hölder Continuity assumption   This is the assumption presented above as Assumption 1. Lipschitzness is a common assumption, and is theoretically grounded for some embeddings (see e.g., Lemma 1 in (Sener & Savarese, 2018) for CNN). Our assumption relaxes Lipschitzness, and our theoretical finding are therefore more general.

In the following, to ease notation for each element $e \in \mathcal{D}$, we will also use e to denote its embedding in $\mathbb{R}^d$.

The problem we consider throughout the rest of the paper is the *Data Selection under well-behaved loss* problem, which

is the problem of Definition 3 when the loss function ℓ is well behaved (see the formal definition in Assumption 1). This definition can be extended to the active learning setting where the objective is to iteratively choose a set of elements to sample based on the model updates.

Definition 4 (r-Adaptive Active learning under well-behaved norm). *The r-adaptive active learning problem under (z, λ)-Hölder Continuity is defined as follows:*

- ***Input:*** *A set of elements $\mathcal{D} \subset \mathbb{R}^d$, oracle access to a function $\ell : \mathcal{D} \mapsto \mathbb{R}_+$ that is well-behaved, a target size s and an adaptivity parameter r.*

- ***Adaptivity:*** *There are r rounds. At round i, the algorithm can query ℓ on a set Q_i of size at most s. Q_i can only be defined based on the results of ℓ on $\cup_{j<i} Q_j$ and $\mathcal{D}$.*

- ***Output:*** *For all $i \in [r]$, a sample $S_i \subseteq \mathcal{D}$ of size at most s together with a weight function $w_i : S \mapsto \mathbb{R}_+$ such that S_i minimizes*

$$\Delta(S) := |\sum_{e \in \mathcal{D}} \ell(e) - \sum_{e \in S_i} w_i(e)\ell(e)|.$$

2.3. Clustering Preliminaries

We defer a detailed description on clustering preliminaries to Appendix A.2. Most importantly, the (k, z)-clustering cost of C on $\mathcal{D}$ is $\Phi_z(\mathcal{D}, C) := \sum_{x \in \mathcal{D}} \min_{c \in C} \|x - c\|^z$ and $\Phi_{k,z}(\mathcal{D}) := \min_{C \subset \mathbb{R}^d, |C| \leq k} \Phi_z(\mathcal{D}, C)$. For $z = 1$, this objective corresponds to k-median, while for $z = 2$ it corresponds to k-means. We call a clustering $\mathcal{C}$ any partition of $\mathcal{D}$ into k parts (called clusters) $C_1, \ldots, C_k$. Given a $\Lambda \in \mathbb{R}^k$, we define $\Phi_{\mathcal{C},z}^{\Lambda}(\mathcal{D}) = \sum_{i=1}^{k} \Lambda_i \Phi_{1,z}(C_i)$.

3. Algorithmic Results

We now study sampling procedures for the active learning problem, defined in the previous section. Our goal is to build a sampling strategy such that $\sum_{s \in S} w(s)\ell(s)$ is an unbiased estimator of $\sum_{e \in \mathcal{D}} \ell(e)$, and show that the estimator is tightly concentrated around its mean. We will first study the case where it is not possible to query the loss function at all, and show a lower bound on the error achievable. We then present some adaptive algorithms, which query the loss function sparingly.

3.1. Algorithm and Lower Bound for the Non-Adaptive Case

We first focus on the context where the algorithm cannot query function ℓ at all. In this case, if one only assumes the loss function to be Hölder continuous the error must scale linearly with both the size of the dataset and the diameter of

its embedding, and this can be achieved by a random sample of the data points, as we show in the following theorem. The proof is deferred to Appendix B.1

Theorem 5. *Let $\varepsilon, \lambda > 0$. There is a constant c, a dataset $\mathcal{D}$ and a loss function ℓ that is $(1, z)$-Hölder such that, when S is a uniform sample of size $1/\varepsilon^2$ with weight function $w(e) = n/s$, it holds with constant probability that*

$$\Delta(S) = |\sum_{e \in \mathcal{D}} \ell(e) - \sum_{e \in S} w(e)\ell(e)| \geq c\varepsilon n \sup_{e \in \mathcal{D}} \ell(e).$$

Furthermore, this lower bound is tight: for all datasets $\mathcal{D}$ and loss functions ℓ, a uniform sample S of size $s = O(1/\varepsilon^2)$ with weights $w(e) = n/s$ satisfies with constant probability $\Delta(S) \leq \varepsilon n \sup_{e \in \mathcal{D}} \ell(e)$.

3.2. Adaptive Algorithms

The lower bound on $\Delta(S)$ in Theorem 5 shows that one must sample more carefully if good guarantees are desired. As explained in the introduction, we present a sampling strategy that queries at most $O(k)$ many points, and reduce the additive error to $\varepsilon\lambda\Phi_k(\mathcal{D})$. This is always better than $\varepsilon n \sup_{e \in \mathcal{D}} |\ell(e)|$, and, in case the embedding of $\mathcal{D}$ has a clustered structured, can be drastically smaller.

3.2.1. 1-ROUND ALGORITHM

In this section we state the 1-round algorithm and show Theorem 2 (with the full proof deferred to Appendix B.2). We start by the pseudo-code of the algorithm.

Algorithm 1 Data-Selection($\mathcal{D}, k, \varepsilon, \Lambda, \mathcal{C}$)

1: **Input**: a dataset $\mathcal{D}$ partitioned into clusters $\mathcal{C} = (C_1, \ldots, C_k)$ with centers $c_1, \ldots, c_k$ and a k-tuple of parameters $\Lambda_1, \ldots, \Lambda_k$.
2: For $e \in C_i$, define $\hat{\ell}(e) := \ell(c_i)$ and $v(e) := \|e - c_i\|^z$.
3: Let $s := \lceil \varepsilon^{-2}(2 + 2\varepsilon/3) \rceil$. For $e \in C_i$ define $p_e := \frac{\hat{\ell}(e) + \Lambda_i v(e)}{\sum_i \Lambda_j \Phi(C_i, \{c_i\}) + \sum_{x \in \mathcal{D}} \hat{\ell}(x)}$ and $w(e) = s^{-1}p_e^{-1}$.
4: Compute a sample S of s points, picked independently following the distribution p_e.
5: **Output:** the set S with weights w.

The proof of Theorem 2 works as follows: we let X_i be the random variable corresponding to the contribution of the i-th sample to the cost. $\sum X_i$ is therefore an unbiased estimator for $\sum_{e \in \mathcal{D}} \ell(e)$: we show that each X_i has a small variance, and apply Bernstein's inequality to conclude. We formalize this argument in Appendix B.2.

Remark 6. *Instead of requiring that, in each cluster, the worst-case $\frac{|\ell(x) - \ell(c_i)|}{\|x - c_i\|^z}$ is bounded, we can allow for a few outliers and require only that $\frac{|\ell(x) - \ell(c_i)|}{\|x - c_i\|^z} \leq \Lambda_i$ for all x but*

a $1/k$-fraction of the probability mass defined in line 3 of Algorithm 1. This allows to have some outliers,

3.2.2. r-ROUND ALGORITHM

We now turn to obtaining better guarantees than the above bounds by allowing for more rounds. Here, we assume that we are given a set of centers $c_1, c_2, \ldots$ such that for all k, the set $(c_1, \ldots, c_k)$ is a good solution to (k, z)-clustering (namely, a solution whose cost is at most $O(1)$ times the optimal cost). Note that this is precisely the guarantee of the k-means++ algorithm (Arthur & Vassilvitskii, 2007) (and, more generally, D^z sampling). We let $\mathcal{C}_k$ be the set of clusters corresponding to the centers $c_1, \ldots, c_k$.

Theorem 7. *Let $\varepsilon > 0, \Lambda \in \mathbb{R}^k$, and integer $r > 0$. Let $\mathcal{D}$ be a dataset with a set of centers $c_1, \ldots, c_{kr}$, and ℓ be be a loss function that is well-behaved with respect to Λ and $\mathcal{C}_i$, for all $i \in \{k, 2k, \ldots, kr\}$. Then there exists an algorithm that for each round $i \in [r]$, queries k elements per round and outputs a sample S_i of size at most $O(1/\varepsilon^2)$ and a weight function w_i such that:*

$$\Delta(S_i) = |\sum_{e \in \mathcal{D}} \ell(e) - \sum_{s \in S_i} w_i(s)\ell(s)|$$

$$\leq \varepsilon \left(\sum_{e \in \mathcal{D}} \ell(e) + \Phi_{\mathcal{C}_{ik}, z}^{\Lambda}(\mathcal{D}) \right)$$

Note that the above algorithm allows to trade-off the round complexity and sample size and reaches optimality in the limit: when $r \cdot k = |\mathcal{D}|$, we obtain an exact algorithm. The algorithm is very similar to Algorithm 1: we defer the presentation to Appendix B.3.

3.2.3. COMPUTING THE CLUSTERING $\mathcal{C}$ AND THE PARAMETER Λ

Our algorithms require the knowledge of a clustering $\mathcal{C}$ and the vector of parameters Λ. We explain here how to compute those values.

Finding a Clustering Our theorems require that the loss is well-behaved w.r.t an estimate of Λ and a clustering $(C_1, \ldots, C_k)$. To compute such a clustering, we can use any algorithm for (k, z)-clustering, e.g., D^z-sampling (the generalization of k-means++ (Arthur & Vassilvitskii, 2007)), or some faster algorithms, e.g. (Cohen-Addad et al., 2020; 2021a).

Estimating Λ. Once we are given a clustering, we can query the loss function in order to estimate Λ. Formally, we have the following:

Lemma 8. *Assume that there is a probability p such that,*

for each cluster C_i,

$$\Pr_{x \in C_i} \left[\frac{|\ell(x) - \ell(c_i)|}{\|x - c_i\|^z} \in [\Lambda_i / \log(n), \Lambda_i] \right] \geq p,$$

where the probability is taken over an x chosen uniformly at random from C_i. Then, one can compute an upper bound on each Λ_i with probability $99/100$ by querying the loss of $\log(100k)/\log(1-p)$ points per cluster.

The proof is deferred to Appendix B.4. We note that we could have made different assumptions in the previous lemma: the $\log(n)$ is picked somewhat arbitrarily, to fit with an exponentially decreasing tail on the distribution of the ratios $\frac{|\ell(x) - \ell(c_i)|}{\|x - c_i\|^z}$. We believe that this assumption is quite natural, and indeed our experiments confirm it across different applications (see Table 1).

4. Data Selection for Regression

In this section, we specialize our method, i.e., sampling according to (k, z)-clustering cost, to the setting of linear regression. Ideally, given a matrix A, our goal is to compute a sketching and rescaling diagonal matrix S with as few non-zero entries as possible such that computing the optimal regression on S is equivalent to computing it on A. For this, we are seeking a "coreset guarantee", namely we want $\|SAx - b\| \approx \|Ax - b\|$, for all x. In the following a_i denotes the i-th row of A. Therefore, we define the data selection problem for regression as the data selection problem of Definition 3, with loss function for the i-th row $\ell(i) = (\langle a_i, x \rangle - b_i)^2$. In that case, we note that the total loss $\sum_{i \in [n]} \ell(i)$ is $\|Ax - b\|_2^2$.

We show in Section 5.1 that our method is faster and provides results as accurate of the state-of-the-art data selection mechanisms. We provide here some theoretical explanations for this success.

As is the case for the previous active-learning problem, we need to make several assumptions, both on A, and b and to restrict the set of possible x. Our first set of assumptions, similar to Assumption 1, is the following:

Assumption 9. *For all $i, \|a_i\|_2 = O(1)$ and $b_i = O(1)$. Given $\Lambda \in \mathbb{R}^k$ and a k-clustering $\mathcal{C} = (C_1, \ldots, C_k)$ of the indices, we say that the input is well-behaved w.r.t Λ and $\mathcal{C}$ when, for every index j in cluster C_i $|b_i - b_j| \leq \Lambda_i \|a_i - a_j\|_2^z$ (where the concatenation of the vectors a_i and $[b_i]$ is the center of cluster C_i).*

The above assumption is similar to the Hölder-continuity assumption we made for Theorem 2: it formalizes that in the sample, the labels (i.e., b_is) must be close to those of their centers when their embeddings (i.e., the a_is) are close. As before, this is necessary to get any result querying a sublinear number of labels b_i. This assumption is also related

to that of (Sener & Savarese, 2018) who assume the labels of the data are drawn randomly, following distributions that are Lipschitz.

The basic idea of our algorithm is to interpret each row of A as a point in $\mathbb{R}^d$ and cluster these points using k-median. Then we compute the optimal regression solution x_0 for the dataset consisting of all centers, each weighted by the size of its cluster, and use x_0 to define a probability distribution over all points. Sampling s points according to this distribution gives a set S together with a suitable weight function.

Algorithm 2 Data-Selection-Regression$(A, k, \varepsilon, \Lambda, \mathcal{C})$

1: **Input:** a matrix A representing the dataset, a clustering $\mathcal{C} = (C_1, \ldots, C_k)$ of the dataset, and a k-tuple of parameters $\Lambda_1, \ldots, \Lambda_k$.
2: For all $i \in [n]$, let j be such that a_j is the center of a_i's cluster: define $\hat{a}_i = a_j$, $\hat{b}_i = b_j$ and the function $v(a_i, x) = (\langle \hat{a}_i, x \rangle - \hat{b}_i)^2$.
3: Compute the optimal regression x_0 for the dataset $\{\hat{a}_1, \ldots, \hat{a}_n\}$, i.e., the dataset where each center of $\mathcal{A}$ is weighted by the size of its cluster.
4: For i in clustering C_j, define $p_i := \frac{\Lambda_j \|a_i - \hat{a}_i\| + v(a_i, x_0)}{\sum_{j' \in [k]} \Lambda_{j'} \Phi(C_{j'}) + \sum_{i' \in [n]} + v(a_{i'}, x_0)}$, and $w(i) = s^{-1} p_i^{-1}$.
5: Compute a sample S of s points, picked independently following the distribution p.
6: **Output:** the set S with weights w.

Our main theorem for regression is stated next. The proof is deferred to Appendix B.5.

Theorem 10. *Let Λ, A and b respect Assumption 9 for a clustering $\mathcal{C}$, with Λ_i being constants, and let $\hat{a}_j$ and x_0 be as computed by Algorithm 2. Let $\delta \in (0, 1/2)$.*

Let $\mathcal{X}$ be the set of vectors x such that $\|x\|_2 = O(1)$ and $\forall j \in C_i, |\langle \hat{a}_j, x - x_0 \rangle| \leq \Lambda_i \|a_j - \hat{a}_i\|_2$. For $s = O(d/\varepsilon^2 \log(1/\delta))$, it holds with probability $1 - \delta$ that, for all $x \in \mathcal{X}$,

$$\left| \sum_{s \in S} w(s)(\langle a_s, x \rangle - b_s)^2 - \|Ax - b\|_2^2 \right| \leq$$
$$\varepsilon(\|Ax - b\|_2^2 + \Phi_{\mathcal{C}, 1}^{\Lambda}(\mathcal{D}))$$

5. Experiments

In Section 5.1 we present experiments on a linear regression task, and in Section 5.2 we present results on neural networks: first for an LLM translation task, and then for image classification.

5.1. Experiments on Linear Regression

Following our theoretical analysis in Section 4, we validate our coreset sampling algorithm on a linear regression
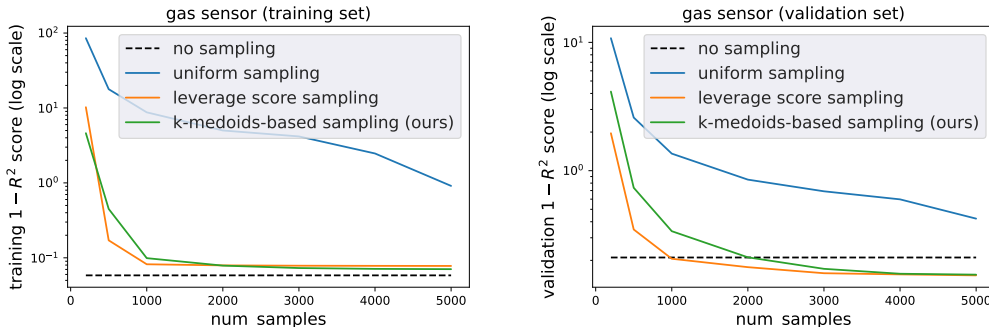
Figure 2. Experimental results on the gas sensor regression dataset. Each data point is the average of 100 runs.

task. We present our results on the UCI gas sensor dataset from the University of California, Irvine repository (Vergara, 2012; Vergara et al., 2012; Rodriguez-Lujan et al., 2014) in Figure 2. The dataset consists of 13910 input points in 16 dimensions. We report the R^2 score, $R^2 := 1 - \frac{\sum_{i=1}^{n}(b_i - x_i)^2}{\sum_{i=1}^{n}(b_i - \overline{y})^2}$, where $\overline{y} := \frac{1}{n}\sum_{i=1}^{n} b_i$. We run Algorithm 2 with some implementation details that are deferred to Appendix C.4.

We compare with uniform sampling and leverage score sampling. Leverage score sampling is the standard sampling algorithm for linear regression, and is known to have extremely good performance but high runtime cost, since it requires solving a full-dimensional linear system. Surprisingly, we find that our clustering-based algorithm performs almost equally well as leverage score sampling, while being drastically faster – the k-medoids solution can be computed in linear time. We report the observed value for Λ across the whole dataset in Table 1.

5.2. Experiments on Neural Networks

5.2.1. FINE-TUNING LARGE LANGUAGE MODELS

We use Algorithm 1 to select a sample on which to fine-tune an LLM for a translation task. We use the WMT T2T EnDe translation task dataset (Bojar et al., 2014) which consists of $4,592,289$ training examples, a test set of size 3003 and a validation set of size 3000. We deduplicated any repeated examples to clean the dataset. We fine-tune a T5-Small model (Raffel et al., 2019) with 77M parameters (details in C.2). To quantify the effect of the quality of the embedding used, we experiment with two different embeddings for the input, (1) BERT (Devlin et al., 2018) and (2) Sentence-T5 (Ni et al., 2021). We run three different methods to subsample approximately 1% of the data (45000 training examples). We ran sensitivity sampling with $k = 4500$ and subsample 45000 elements of the data and a Hölder constant of 0.1 [2]. The diversity sampling methods resembles the one

---

[2]We also experimented with Hölder constants up to 500. See C.2 for more details.

of (Sener & Savarese, 2018): it consists of running k-means (instead of k-center), with $k = 45000$ and using the elements closest to centers for training. Random is a uniform sample of the dataset of size 45000. Random-deduped is a uniform sample as well, ensuring no duplicates. We show that our methods drastically improve over uniform or diversity and observe that the results are consistent across the two types of embeddings used (see Figure 3). In particular, we note: fine-tuning on the full dataset for $100,000$ steps yields an evaluation accuracy of 0.7; as such, we improve from a 0.59 random baseline to 0.6; covering 9% of the headroom.

5.2.2. DATA SELECTION FOR IMAGE CLASSIFICATION

For our classification experiments, our setting is as follows: given a target number of data points k that we need to sample, we first train an initial model $\mathcal{M}$ using a uniformly random subset of $k' < k$ data points, and then sample the remaining $k - k'$ data points using Algorithm 1 with the loss defined by $\mathcal{M}$. Finally, we train a model on all the k data points and evaluate it on a validation set. For our experiments, we chose $k' = 0.2k$. We use three classic datasets from UCI, MNIST (LeCun et al., 1998), FMNIST (Xiao et al., 2017), and CIFAR-10 (Krizhevsky et al., 2009).

Our algorithms. We consider two instantiations of the sensitivity sampling algorithm presented in Algorithm 1: *loss-based sampling* and *gradient-based sampling*. For loss-based sampling, we set $\ell(e) := L(y, \text{model}_\theta(e))$ to be the *loss* of the model on example e with respect to the true label y, where θ are the model parameters. For gradient-based sampling, we set $\ell(e) := \|\nabla_\theta L(y, \text{model}_\theta(e))\|_2^2$ to be equal to the squared ℓ_2 norm of the gradient update.

We present our results in Table 2 and Figure 6 (in Appendix). We notice that the loss- and gradient-based sampling versions of Algorithm 1 perform best when the number of samples is relatively small. In addition, based on the runtime comparison in Figure 6, the loss-based algorithm performs best in terms of runtime.
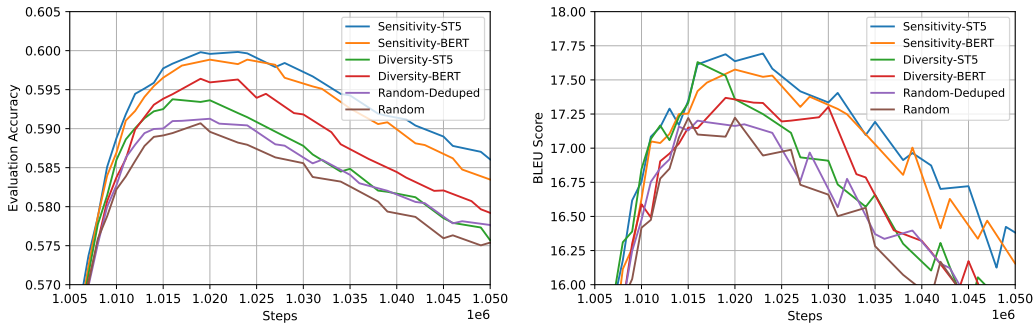
Figure 3. Experimental results on the WMT T2T EnDe translation task dataset. We report the accuracy (left) and BLEU score (right) of the different methods used: Our method (Sensitivity) compared to Diversity (similar to (Sener & Savarese, 2018)), Uniform cleaned (Random-Deduped), and Uniform (Random). Each method is required to produce a sample of roughly 1% of the whole dataset.

Algorithm	MNIST	Fashion MNIST	CIFAR10
uniform	0.9130	0.8091	0.4587
coreset [SS18]	0.9134	0.7692	0.4491
Loss Alg 1	0.9203	**0.8140**	0.4590
Grad Alg 1	**0.9207**	0.8107	**0.4598**

Table 2. Experimental results for selecting $k = 2000$ data points and different datasets. For each algorithm, we show the accuracy on the validation dataset, averaged over 100 runs.
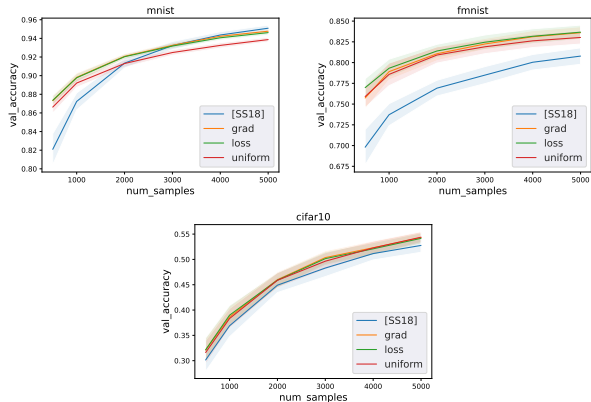
Figure 4. Plots of experimental results for different datasets. For each algorithm, we plot the accuracy on the validation dataset for different values of k (number of samples). We also provide a runtime comparison on CIFAR10. We independently run each data point 100 times, and present the mean with bands of one standard deviation.

Acknowledgments

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Amin, K., Cortes, C., DeSalvo, G., and Rostamizadeh, A. Understanding the effects of batching in online active learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3482–3492. PMLR, 2020.

Arthur, D. and Vassilvitskii, S. k-means++: the advantages of careful seeding. In Bansal, N., Pruhs, K., and Stein, C. (eds.), *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007,*

pp. 1027–1035. SIAM, 2007. URL http://dl.acm.org/citation.cfm?id=1283383.1283494.

Bachem, O., Lucic, M., and Krause, A. Scalable k-means clustering via lightweight coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1119–1127, 2018.

Balcan, M.-F., Broder, A., and Zhang, T. Margin based active learning. In *International Conference on Computational Learning Theory*, pp. 35–50. Springer, 2007.

Bansal, N., Cohen-Addad, V., Prabhu, M., Saulpic, D., and Schwiegelshohn, C. Sensitivity sampling for k-means: Worst case and stability optimal coreset bounds, 2024.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L., and Tamchyna, A. s. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pp. 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3302.

Brinker, K. Incorporating diversity in active learning with support vector machines. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 59–66, 2003.

Chakraborty, S., Balasubramanian, V., and Panchanathan, S. Adaptive batch mode active learning. *IEEE transactions on neural networks and learning systems*, 26(8):1747–1760, 2014.

Chen, X. and Derezinski, M. Query complexity of least absolute deviation regression via robust uniform convergence. In Belkin, M. and Kpotufe, S. (eds.), *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pp. 1144–1179. PMLR, 2021.

Chen, X. and Price, E. Active regression via linear-sample sparsification. In Beygelzimer, A. and Hsu, D. (eds.), *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, volume 99 of *Proceedings of Machine Learning Research*, pp. 663–695. PMLR, 2019.

Citovsky, G., DeSalvo, G., Gentile, C., Karydas, L., Rajagopalan, A., Rostamizadeh, A., and Kumar, S. Batch active learning at scale. *Advances in Neural Information Processing Systems*, 34, 2021.

Cohen-Addad, V., Lattanzi, S., Norouzi-Fard, A., Sohler, C., and Svensson, O. Fast and accurate k-means++ via rejection sampling. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Cohen-Addad, V., Lattanzi, S., Norouzi-Fard, A., Sohler, C., and Svensson, O. Parallel and efficient hierarchical k-median clustering. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 20333–20345, 2021a.

Cohen-Addad, V., Saulpic, D., and Schwiegelshohn, C. A new coreset framework for clustering. In *Symposium on Theory of Computing, STOC*, pp. 169–182, 2021b. doi: 10.1145/3406325.3451022. URL https://doi.org/10.1145/3406325.3451022.

Cohen-Addad, V., Saulpic, D., and Schwiegelshohn, C. Improved coresets and sublinear algorithms for power means in euclidean spaces. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 21085–21098, 2021c.

Cohen-Addad, V., Larsen, K. G., Saulpic, D., and Schwiegelshohn, C. Towards optimal lower bounds for k-median and k-means coresets. In Leonardi, S. and Gupta, A. (eds.), *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pp. 1038–1051. ACM, 2022a. doi: 10.1145/3519935.3519946. URL https://doi.org/10.1145/3519935.3519946.

Cohen-Addad, V., Larsen, K. G., Saulpic, D., Schwiegelshohn, C., and Sheikh-Omar, O. A. Improved coresets for euclidean k-means. In *NeurIPS*, 2022b.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

Dasgupta, S. Analysis of a greedy active learning strategy. *Advances in neural information processing systems*, 17, 2004.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Elhamifar, E., Sapiro, G., Yang, A., and Sasrty, S. S. A convex optimization framework for active learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 209–216, 2013.

Esfandiari, H., Karbasi, A., and Mirrokni, V. S. Adaptivity in adaptive submodularity. In Belkin, M. and Kpotufe, S. (eds.), *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pp. 1823–1846. PMLR, 2021.

Feldman, D. and Langberg, M. A unified framework for approximating and clustering data. In *Symposium on Theory of Computing, STOC*, pp. 569–578, 2011.

Golovin, D. and Krause, A. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

Gonen, A., Sabato, S., and Shalev-Shwartz, S. Efficient active learning of halfspaces: an aggressive approach. In Dasgupta, S. and McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 480–488, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/gonen13.html.

Guillory, A. and Bilmes, J. Interactive submodular set cover. *arXiv preprint arXiv:1002.3345*, 2010.

Guo, Y. and Schuurmans, D. Discriminative batch mode active learning. In *NIPS*, pp. 593–600. Citeseer, 2007.

Hanneke, S. A bound on the label complexity of agnostic active learning. In *Proceedings of the 24th international conference on Machine learning*, pp. 353–360, 2007.

Har-Peled, S. and Mazumdar, S. On coresets for k-means and k-median clustering. In *Symposium on Theory of Computing, STOC*, pp. 291–300, 2004.

Hochbaum, D. S. and Pathria, A. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics (NRL)*, 45(6):615–627, 1998.

Hoi, S. C., Jin, R., Zhu, J., and Lyu, M. R. Batch mode active learning and its application to medical image classification. In *Proceedings of the 23rd international conference on Machine learning*, pp. 417–424, 2006.

Huang, L., Li, J., and Wu, X. On optimal coreset construction for euclidean (k, z)-clustering, 2023.

Joshi, A. J., Porikli, F., and Papanikolopoulos, N. Multiclass active learning for image classification. In *2009 ieee conference on computer vision and pattern recognition*, pp. 2372–2379. IEEE, 2009.

Joshi, A. J., Porikli, F., and Papanikolopoulos, N. Multiclass batch-mode active learning for image classification. In *2010 IEEE international conference on robotics and automation*, pp. 1873–1878. IEEE, 2010.

Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T. Active learning with gaussian processes for object categorization. In *2007 IEEE 11th international conference on computer vision*, pp. 1–8. IEEE, 2007.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Littlewood, J. E. and Offord, A. C. On the number of real roots of a random algebraic equation. ii. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 35, pp. 133–148. Cambridge University Press, 1939.

Maalouf, A., Jubran, I., and Feldman, D. Fast and accurate least-mean-squares solvers for high dimensional data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44 (12):9977–9994, 2022. doi: 10.1109/TPAMI.2021. 3139612. URL https://doi.org/10.1109/TPAMI.2021.3139612.

Mettu, R. R. and Plaxton, C. G. The online median problem. *SIAM J. Comput.*, 32(3):816–832, 2003. doi: 10.1137/S0097539701383443. URL https://doi.org/10.1137/S0097539701383443.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Musco, C., Musco, C., Woodruff, D. P., and Yasuda, T. Active linear regression for ℓ_p norms and beyond. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pp. 744–753. IEEE, 2022.

Mussay, B., Osadchy, M., Braverman, V., Zhou, S., and Feldman, D. Data-independent neural pruning via coresets. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gmHaEKwB.

Ni, J., Ábrego, G. H., Constant, N., Ma, J., Hall, K. B., Cer, D., and Yang, Y. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*, 2021.

Parulekar, A., Parulekar, A., and Price, E. L1 regression with lewis weights subsampling. In Wootters, M. and Sanità, L. (eds.), *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pp. 49:1–49:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, 2019. URL http://arxiv.org/abs/1910.10683.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B. B., Chen, X., and Wang, X. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

Rodriguez-Lujan, I., Fonollosa, J., Vergara, A., Homer, M., and Huerta, R. On the calibration of sensor arrays for pattern recognition using the minimal number of experiments. *Chemometrics and Intelligent Laboratory Systems*, 130:123–134, 2014.

Roy, N. and McCallum, A. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, 2:441–448, 2001.

Schubert, E. and Rousseeuw, P. J. Faster k-medoids clustering: improving the pam, clara, and clarans algorithms. In *Similarity Search and Applications: 12th International Conference, SISAP 2019, Newark, NJ, USA, October 2–4, 2019, Proceedings 12*, pp. 171–187. Springer, 2019.

Sener, O. and Savarese, S. Active learning for convolutional neural networks: A core-set approach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=H1aIuk-RW.

Settles, B. Active learning literature survey. 2009.

Tong, S. and Koller, D. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

Tukan, M., Baykal, C., Feldman, D., and Rus, D. On coresets for support vector machines. In Chen, J., Feng, Q., and Xu, J. (eds.), *Theory and Applications of Models of Computation, 16th International Conference, TAMC 2020, Changsha, China, October 18-20, 2020, Proceedings*, volume 12337 of *Lecture Notes in Computer Science*, pp. 287–299. Springer, 2020a.

doi: 10.1007/978-3-030-59267-7_25. URL https://doi.org/10.1007/978-3-030-59267-7_25.

Tukan, M., Maalouf, A., and Feldman, D. Coresets for near-convex functions. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 997–1009. Curran Associates, Inc., 2020b. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/0afe095e81a6ac76ff3f69975cb3e7ae-Paper.pdf.

Tukan, M., Mualem, L., and Maalouf, A. Pruning neural networks via coresets and convex geometry: Towards no assumptions. In *NeurIPS*, 2022.

Tukan, M., Zhou, S., Maalouf, A., Rus, D., Braverman, V., and Feldman, D. Provable data subset selection for efficient neural networks training. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 34533–34555. PMLR, 2023. URL https://proceedings.mlr.press/v202/tukan23a.html.

Vergara, A. Gas Sensor Array Drift Dataset. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5RP6W.

Vergara, A., Vembu, S., Ayhan, T., Ryan, M. A., Homer, M. L., and Huerta, R. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 166:320–329, 2012.

Wang, Z. and Ye, J. Querying discriminative and representative samples for batch mode active learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9 (3):1–23, 2015.

Wei, K., Iyer, R., and Bilmes, J. Submodularity in data subset selection and active learning. In *ICML*, pp. 1954–1963. PMLR, 2015.

Woodruff, D. P. and Yasuda, T. New subset selection algorithms for low rank approximation: Offline and online. In Saha, B. and Servedio, R. A. (eds.), *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pp. 1802–1813. ACM, 2023.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Yang, Y., Ma, Z., Nie, F., Chang, X., and Hauptmann, A. G. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113:113–127, 2015.

Yu, K., Bi, J., and Tresp, V. Active learning via transductive experimental design. In *Proceedings of the 23rd international conference on Machine learning*, pp. 1081–1088, 2006.

A. Preliminaries

A.1. Further Related Work

We present related work in data selection and active learning. For an extensive and recent survey, we refer to (Ren et al., 2021), and detail here some of the most relevant points for comparison.

Our work departs from previous work in the following ways: we work in the regime where we have a budget on both the number of elements selected (i.e., labeled) and the number of inferences of the model. Moreover, we present a rigorous analysis of our sampling mechanism that works for a variety of machine learning models, as long as we are provided with embeddings that are Lipshitz with respect to the loss of the model.

Given an unlabeled set of points, the active learning problem asks to identify the most relevant points to label ((Settles, 2009; Cohn et al., 1996)). In the era of big data, labeling a big dataset is often too expensive. We are thus given a budget of, say, k elements that we can label. The question becomes how to pick these k elements so as to maximize the performance of the final model (that will be trained on these elements).

From a theoretical standpoint, (Dasgupta, 2004) demonstrated that greedy active learning heuristics perform poorly if agnostic to both data and learning algorithm. To circumvent these negative results, other works have made assumptions on the data-dependent realizability of the hypothesis space like ((Gonen et al., 2013)) or on a data dependent measure of the concept space called disagreement coefficient ((Hanneke, 2007)).

Related to ours, several successful works have brought together unsupervised techniques such as clustering and information from the model (such as margin scores), see e.g.: (Citovsky et al., 2021). The work of (Sener & Savarese, 2018) brings together clustering, and sketching techniques (coresets in this case).

Another line of works consists of Bayesian active learning methods which use a non-parametric model, such as a Gaussian process, to obtain an estimate of the expected improvement on the model after each query ((Kapoor et al., 2007)), or alternatively the expected error after a set of queries ((Roy & McCallum, 2001)). It seems that an important drawback is that these approaches do not scale to large models (see the discussion in (Sener & Savarese, 2018)).

Uncertainty based methods form another important family of active learning algorithm. They aim at finding relevant examples using heuristics like highest entropy ((Joshi et al., 2009)), or geometric distance to decision boundaries ((Tong & Koller, 2001; Brinker, 2003)).

Batch-active learning based on uncertainty may lead to a useless batch, where all queries are very similar (when the highest uncertainty is concentrated in a small region). To cope with this, several methods aim at trading-off diversity and uncertainty to select the points. The way the elements are iteratively selected can vary depending on the application from mini-batches to one-shot (e.g., (Hoi et al., 2006; Guo & Schuurmans, 2007; Chakraborty et al., 2014; Citovsky et al., 2021; Amin et al., 2020)) Specifically in the context of mini-batch active learning, a common approach is to use unsupervised machine learning techniques to extract information from the data. Such methods include k- Medoid ((Schubert & Rousseeuw, 2019)), or MaxCover ((Hochbaum & Pathria, 1998)) to select a set of data points that maximally cover the dataset with respect to some objective. (Elhamifar et al., 2013) and (Yang et al., 2015) design a discrete optimization problem for this purpose, that they solve using convex optimization methods. Unfortunately, the running time of these methods is quadratic in the input data size and so highly impractical for large data.

Covering or clustering approaches have also been tried in the past (Joshi et al., 2010; Wang & Ye, 2015). The former does not provide any theoretical guarantee associated to its approach. The latter uses empirical risk minimization to minimize the difference between the maximum mean discrepancy between i.i.d. samples from the dataset and the actively selected samples (instead of the loss we work with).

Active learning has also been studied when tailored to some specific machine learning problems such as nearest neighbors, logistic regression or linear regression with Gaussian noise ((Wei et al., 2015; Hoi et al., 2006; Guo & Schuurmans, 2007; Yu et al., 2006)).

Recently, active learning was also extended to ℓ_p-regression for all $p \geq 1$ without any assumptions on the data, resulting in a number of optimal bounds (Chen & Price, 2019; Chen & Derezinski, 2021; Parulekar et al., 2021; Musco et al., 2022; Woodruff & Yasuda, 2023). These works are based on using sampling probabilities defined from the design matrix (agnostic to the label vector), and range from leverage scores ($p = 2$) to ℓ_p-sensitivities to ℓ_p-Lewis weights, the latter achieving optimal bounds for all $p \geq 1$. Our work adds a new set of scores to this growing literature for regression, namely, scores that

are proportional to the cost of clustering individual points. We note that in practice, computing an approximate k-median solution may be much faster than approximating the Lewis weights or leverage scores of a matrix since it does not involve computing the inverse of any matrices.

If the model can be run on all input data, and so the confidence of the model is known for all the input data elements, then several set-cover based methods that aim at best covering the hypothesis space have been designed ((Guillory & Bilmes, 2010; Golovin & Krause, 2011; Esfandiari et al., 2021). The key distinguishing factor of our approach compared to these works is that we do not require the model to be run on all the input data. Furthermore, as we demonstrate, our sampling technique applies more generally to problems other than regression.

Coresets Our ideas are inspired from the coreset literature. Coreset were introduced initially for k-median and k-means clustering: the goal is to compute a (weighted) set S of points such that, for any set of k centers, evaluating its cost on S is almost the same as evaluating it on the full dataset (Har-Peled & Mazumdar, 2004). Coresets with optimal size (which is $O(k\varepsilon^{-2}\min(\varepsilon^{-2}, \sqrt{k}))$ exist (Cohen-Addad et al., 2021b; 2022a;b; Huang et al., 2023), and one of the most standard tools to build a coreset is sensitivity sampling, namely sampling according to the cost in a constant-factor (k, z)-clustering solution.

Ideas from the literature on coreset for clustering have already spread to other domains: (Tukan et al., 2023) presents coresets for radial basis function neural networks of small sizes, and (Tukan et al., 2020b) for near-convex functions. (Mussay et al., 2020; Tukan et al., 2022) showed how to use coresets for prunning and compressiong neural networks, and (Maalouf et al., 2022) used coresets for fast least-square linear regression. For machine-learning tasks, (Tukan et al., 2020a) present coresets for Support Vector Machines.

A.2. Clustering Preliminaries

In the following, we are given a set of points $\mathcal{D}$ in the Euclidean Space $\mathbb{R}^d$ with ℓ_2 norm. We let $\mu_z(\mathcal{D})$ be the power mean of X, namely the point p that minimizes $\sum_{x \in \mathcal{D}} \|x - p\|^z$. We let $\mathrm{Disp}_z(\mathcal{D}) := \sum_{x \in X} \|x - \mu(\mathcal{D})\|^z$.

Given a set of k points $C \in (\mathbb{R}^d)^k$, we denote the (k, z)-clustering cost of C on $\mathcal{D}$ as $\Phi_z(\mathcal{D}, C) := \sum_{x \in \mathcal{D}} \min_{c \in C} \|x - c\|^z$ and $\Phi_{k,z}(\mathcal{D}) := \min_{C \subset \mathbb{R}^d, |C| \le k} \Phi_z(\mathcal{D}, C)$. For $z = 1$, this objective corresponds to k-median, while k-means is for $z = 2$. Note that $\mathrm{Disp}_z(\mathcal{D}) = \Phi_{1,z}(\mathcal{D})$.

We say that a set C of k points is an α-approximation to (k, z)-clustering on $\mathcal{D}$ when $\Phi_z(C, \mathcal{D}) \le \alpha\Phi_{k,z}(\mathcal{D})$. An ordered list of centers $c_1, \ldots, c_n$ is an α-approximation to Prefix-z-clustering when, for all $1 \le k \le n$, $(c_1, \ldots, c_k)$ is an α-approximation to (k, z)-clustering on $\mathcal{D}$. An $O(1)$-approximation to prefix (k, z)-clustering can be computed using the algorithm of (Mettu & Plaxton, 2003). D^z-sampling (which is k-means++ for $z = 2$) gives an $O(\log k)$-approximation, which is fast and performs extremely well in practice.

B. Deferred Proofs

In this section, we use Bernstein's concentration inequality:

Theorem 11 (Bernstein's inequality). *Let $X_1, \ldots, X_n$ be independent random variables and let $M > 0$ be such that, for all i, $|X_i| \le M$. Then, for all $t > 0$,*

$$Pr[|\sum_i X_i - E[\sum_i X_i]| \ge t] \le \exp(-\frac{t^2}{2\sum_{x \in X} E[X_x^2] + 2Mt/3})$$

B.1. Proof for the non-adaptive case

Proof of Theorem 5. We denote for simplicity $R = \sup_{e \in \mathcal{D}} \ell(e)$. The upper bound is a simple application of Bernstein's inequality and is included for completeness.

The algorithm chooses successively s uniformly random samples $S_1, \ldots, S_s$ from $\mathcal{D}$ (with replacement) and gives each sampled element weight n/s. Let $X_i = w(S_i)\ell(S_i)$. It holds that $\mathbb{E}[X_i] = \frac{n}{s}\sum_{e \in \mathcal{D}} \frac{\ell(e)}{n} = \frac{\sum_{e \in \mathcal{D}} \ell(e)}{s}$, and, thus, $\mathbb{E}[\sum_{e \in S} w(e)\ell(e)] = \mathbb{E}[\sum X_i] = \sum_{e \in \mathcal{D}} \ell(e)$.

To show that this sum of random variables is concentrated, we aim at applying Bernstein's inequality. For this, we need to

bound $\mathbb{E}[X_i^2]$: we have

$$\mathbb{E}[X_i^2] = \sum_{e \in \mathcal{D}} \left(\frac{n}{s} \ell(e) \right)^2 \Pr[e = S_i] \leq \sum_{e \in \mathcal{D}} \frac{n}{s^2} \ell(e)^2 \leq \frac{n^2}{s^2} R^2.$$

Summed over all i, we therefore have $\sum_{i=1}^s \mathbb{E}[X_i^2] \leq n^2 R^2/s$. Furthermore, for any i, $|X_i| \leq \frac{n}{s} R$ with probability 1. Plugging this result into Bernstein's inequality yields

$$\Pr[\Delta(S) \geq \varepsilon n R] = \Pr\left[\left| \sum_i X_i - E[\sum_i X_i] \right| \geq \varepsilon n R \right] \leq \exp\left(-\frac{\varepsilon^2 n^2 R^2 \cdot s}{2n^2 R^2 + 2nR \cdot \varepsilon n R/3} \right)$$
$$\leq \exp(-\varepsilon^2 \cdot s/(2 + \varepsilon)).$$

With $s = O(1/\varepsilon^2)$, this gives the desired probability bound.

For the lower bound, consider a multiset $\mathcal{D} \subset \mathbb{R}$ with $n/2$ copies of -1 and $n/2$ copies of 1, and ℓ being the identify function $\ell(x) = x$, implying that $\sum_{e \in \mathcal{D}} \ell(e) = 0$. Then, the estimator is a sum of Rademacher random variables, and an anti-concentration bound states that for any fixed value x, $\sum_{s \in S} \ell(s) = x$ with probability at most $O(1/\sqrt{|S|})$ (Littlewood & Offord, 1939). Therefore, for some constant c (which depends on the previous big-O constant), $\Pr[|\sum_{s \in S} \ell(s)| \geq c\sqrt{|S|}] \geq 1/2$. Multiplying with $n/|S|$, this implies that our estimator is bigger than $\frac{cn}{\sqrt{|S|}}$ with probability at least 1/2.

When $|S| \leq 1/\varepsilon^2$, this gives the desired statement. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \square$

B.2. Proof of Theorem 2

Proof. We prove that Algorithm 1 yields the desired result. Let $S_1, \ldots, S_s$ be the successive random samples, and define $X_i = \ell(S_i)w(S_i)$.

We combine Bernstein's inequality (see Theorem 11) with Assumption 1 to bound the value of t. Recall that $\Phi_{\mathcal{C},z}^{\Lambda}(\mathcal{D}) := \sum_{i=1}^k \Lambda_i \Phi_z(C_i, c_i)$; and when $e \in C_i$ we let $\Lambda_e = \Lambda_i$

From Assumption 1, we get that for $e \in C_i$, $\ell(e) \leq \hat{\ell}(e) + \Lambda_i v(e)$ and $\hat{\ell}(e) \leq \ell(e) + \Lambda_i v(e)$. Therefore,

$$E[X_i^2] = \sum_{e \in \mathcal{D}} (\ell(e)w(e))^2 \cdot p_e = \sum_{e \in \mathcal{D}} \ell(e)^2 \frac{1}{p_e s^2}$$
$$= \frac{1}{s^2} \cdot \sum_{e \in \mathcal{D}} \ell(e)^2 \cdot \frac{\Phi_{\mathcal{C},z}^{\Lambda}(\mathcal{D}) + \sum_{x \in \mathcal{D}} \hat{\ell}(x)}{\hat{\ell}(e) + \Lambda_e v(e)}$$
$$\leq \frac{1}{s^2} \cdot \sum_{e \in \mathcal{D}} \ell(e) \cdot (\hat{\ell}(e) + \Lambda_e v(e)) \cdot \frac{\Phi_{\mathcal{C},z}^{\Lambda}(\mathcal{D}) + \sum_{x \in \mathcal{D}} \hat{\ell}(x)}{\hat{\ell}(e) + \Lambda_e v(e)}$$
$$= \frac{1}{s^2} \cdot \sum_{e \in \mathcal{D}} \ell(e) \cdot \left(\Phi_{\mathcal{C},z}^{\Lambda}(\mathcal{D}) + \sum_{x \in \mathcal{D}} \hat{\ell}(x) \right)$$
$$\leq \frac{1}{s^2} \cdot \left(\sum_{e \in \mathcal{D}} \ell(e) \right) \cdot \left(2\Phi_{\mathcal{C},z}^{\Lambda}(\mathcal{D}) + \sum_{x \in \mathcal{D}} \ell(x) \right)$$
$$\leq \frac{1}{s^2} \cdot \left(2\Phi_{\mathcal{C},z}^{\Lambda}(\mathcal{D}) + \sum_{e \in \mathcal{D}} \ell(e) \right)^2.$$

We also let $M := \max_{e \in \mathcal{D}} \ell(e)w(e) = \max_e \ell(e) \cdot \frac{\Phi_{\mathcal{C},z}^{\Lambda}(\mathcal{D}) + \sum_x \hat{\ell}(x)}{s(\hat{\ell}(e) + \Lambda_e v(e))}$. Similarly, we have $M \leq \frac{1}{s} \left(\Phi_{\mathcal{C},z}^{\Lambda}(\mathcal{D}) + \sum_{e \in \mathcal{D}} \hat{\ell}(e) \right) \leq \frac{1}{s} \cdot \left(2\Phi_{\mathcal{C},z}^{\Lambda}(\mathcal{D}) + \sum_{e \in \mathcal{D}} \ell(e) \right)$.

16

Applying Berstein's inequality with $t = 2\Phi^{\Lambda}_{\mathcal{C},z}(\mathcal{D}) + \sum_{e \in \mathcal{D}} \ell(e)$ therefore yields:

$$\Pr\left[\Delta(S) > \varepsilon t\right] \le \exp\left(-\varepsilon^2 \frac{t^2 \cdot s}{2t^2 + 2\varepsilon t^2/3}\right) = \exp\left(-\varepsilon^2 \frac{s}{2 + 2\varepsilon/3}\right) \le \exp(-1),$$

where the last inequality holds by the choice of $s = \lceil \varepsilon^{-2}(2 + 2\varepsilon/3) \rceil$. $\qquad\square$

B.3. Adaptive Active learning

We present here the algorithm used to prove Theorem 7. The proof follows directly from the proof for Theorem 2.

Algorithm 3 Adaptive-active-learning($\mathcal{D}, r, \lambda, \varepsilon$)

1: Compute a $O(1)$-approximation to prefix-z-clustering on $\mathcal{D}$, i.e., an ordering of the points in $\mathcal{D}$ such that any prefix of length k_0 is an $O(1)$-approximation to (k_0, z)-clustering on $\mathcal{D}$.
2: **for** each round $i = 1, \ldots, r$ **do**
3:      query ℓ for the points at position in $[(i-1)k+1, ik]$ in the ordering, and define $\mathcal{A}$ to be the set of elements at position at most ik in the ordering.
4:      For $e \in \mathcal{D}$, define $\mathcal{A}(e) = \operatorname{argmin}_{a \in \mathcal{A}} \|e - a\|$ the element of $\mathcal{A}$ that is the closest to x, $\hat{\ell}(e) := \ell(\mathcal{A}(e))$ and $v(e) := \|e - \mathcal{A}(x)\|$.
5:      Define $p_e := \frac{\hat{\ell}(e) + \lambda v(e)}{\lambda \Phi_z(\mathcal{D}, \mathcal{A}) + \sum_{x \in \mathcal{D}} \hat{\ell}(x)}$.
6:      Let $s := \lceil \varepsilon^{-2}(2 + 2\varepsilon/3) \rceil$. For $e \in \mathcal{D}$ define $p_e := \frac{\hat{\ell}(e) + \lambda v(e)}{\lambda \Phi_z(\mathcal{D}, \mathcal{A}) + \sum_{x \in \mathcal{D}} \hat{\ell}(x)}$ and $w_i(e) = s^{-1} p_e^{-1}$.
7:      Compute a sample S_i of s points, picked independently following the distribution p_e.
8: **end for**
9: **Output:** S_i, w_i for each round i.

B.4. The parameter Λ

Proof of Lemma 8. We have, for t points $x_1, \ldots, x_t$ uniformly at random in C_i:

$$\Pr\left[\max_j \frac{|\ell(x_j) - \ell(c_i)|}{\|x_j - c_i\|^z} \in [\Lambda_i / \log(n), \Lambda_i]\right] \ge 1 - (1 - p)^t.$$

With $t = \log(100k)/\log(1 - p)$, the maximum of the estimate $\frac{|\ell(x_j) - \ell(c_i)|}{\|x_j - c_i\|^z}$ is in $[\Lambda_i / \log(n), \Lambda_i]$ with probability at least $1 - 1/(100k)$. Multiplying $\max_j \frac{|\ell(x_j) - \ell(c_i)|}{\|x_j - c_i\|^z}$ by $\log(n)$ thus yields our upper bound on Λ_i. $\qquad\square$

B.5. Regression

To show Theorem 10, we first prove that for any fixed $x \in \mathcal{X}$, the desired bound hold with probability $1 - \exp(-\varepsilon^2 s)$. It is standard to extend this result to all $x \in \mathcal{X}$, using discretization techniques to find a set N of size $\varepsilon^{-O(d)}$ such that preserving the cost for all vectors in N is enough to extend the result for all candidate x (N is called a net for $\mathcal{X}$). Hence, it is enough to show the following lemma:

Lemma 12. *Let $\Lambda \in \mathbb{R}^k$ with $\lambda_i \ge 1$, and A and b that respects Assumption 9 with constant ζ and $\hat{a}_i$ and x_0 as computed by Algorithm 2.*

Let $x \in \mathcal{X}$, namely x such that $\|x\|_2 = O(1)$ and there is some $\zeta \ge 1$ such that $\forall j \in C_i, |\langle \hat{a}_j, x - x_0 \rangle| \le \Lambda_i \|a_j - \hat{a}_j\|_2$. Then, with probability $1 - \delta$, it holds that for $s = 8\varepsilon^{-2}\log(1/\delta)$,

$$\left| \sum_{s \in S} w(s)(\langle a_s, x \rangle - b_s)^2 - \|Ax - b\|_2^2 \right| \le \varepsilon\left(\|Ax - b\|_2^2 + \sum_{i \in [k]} \Lambda_i \Phi_{1,1}(C_i)\right)$$

Proof. Let $S_1, \ldots, S_s$ be the successive random samples, and define $X_t = w(S_t)(\langle a_{S_t}, x \rangle - b_{S_t})^2$. By the choice of w, it holds that $\mathbb{E}[\sum X_t] = \|Ax - b\|_2^2$. We will show concentration using the Bernstein inequality.

We denote for simplicity $\Phi_\Lambda(A) = \sum_{i \in [k]} \Lambda_i \Phi_{1,1}(C_i)$. For $j \in C_i$, we let $\tilde{\Lambda}_j = \Lambda_i$.

We first focus on bounding the second moment of X_t. We have:

$$\mathbb{E}[X_t^2] = \sum_{i=1}^{n} \frac{(\langle a_i, x \rangle - b_i)^4}{s^2 p_i}$$

$$= \frac{1}{s^2} \cdot \sum_{i=1}^{n} (\langle a_i, x \rangle - b_i)^4 \cdot \frac{\sum_{j \in [n]} \tilde{\Lambda}_j \|a_j - \hat{a}_j\| + v(a_j, x_0)}{\tilde{\Lambda}_i \|a_i - \hat{a}_i\| + v(a_i, x_0)}$$

To bound this term, we first note that

$$(\langle a_i, x \rangle - b_i)^2 - (\langle \hat{a}_i, x \rangle - \hat{b}_i)^2$$
$$= (\langle a_i + \hat{a}_i, x \rangle - b_i - \hat{b}_i)(\langle a_i - \hat{a}_i, x \rangle - b_i + \hat{b}_i)$$
$$= \langle a_i + \hat{a}_i, x \rangle \langle a_i - \hat{a}_i, x \rangle + \langle a_i + \hat{a}_i, x \rangle \cdot (\hat{b}_i - b_i) - (b_i + \hat{b}_i)\langle a_i - \hat{a}_i, x \rangle - (\hat{b}_i + b_i)(\hat{b}_i - b_i)$$
$$\leq \|a_i + \hat{a}_i\| \cdot \|a_i - \hat{a}_i\| \cdot \|x\|^2 + \|a_i + \hat{a}_i\| \cdot \|x\||\hat{b}_i - b_i| - |b_i + \hat{b}_i| \cdot \|a_i - \hat{a}_i\| \cdot \|x\| + |b_i + \hat{b}_i| \cdot |b_i - \hat{b}_i|$$
$$= O(\tilde{\Lambda}_i \|a_i - \hat{a}_i\|), \tag{2}$$

where the last two lines follow from Cauchy-Schwarz and Assumption 9.

We now relate this to the term $v(a_i, x_0) = (\langle \hat{a}_i, x_0 \rangle - \hat{b}_i)^2$ of the denominator:

$$(\langle \hat{a}_i, x \rangle - \hat{b}_i)^2 - (\langle \hat{a}_i, x_0 \rangle - \hat{b}_i)^2 = \langle \hat{a}_i, x - x_0 \rangle \cdot (\hat{a}_i, x - x_0) + 2\hat{b}_i)$$
$$= \langle \hat{a}_i, x + x_0 \rangle \langle \hat{a}_i, x - x_0 \rangle - 2\hat{b}_i \langle \hat{a}_i, x - x_0 \rangle$$
$$= O(|\langle \hat{a}_i, x - x_0 \rangle|) = O(\tilde{\Lambda}_i \|a_i - \hat{a}_i\|),$$

where the last line uses our assumption $|\langle \hat{a}_i, x - x_0 \rangle| \leq \tilde{\Lambda}_i \|a_i - \hat{a}_i\|$. Thus, combining those equations:

$$(\langle a_i, x \rangle - b_i)^2 \leq O(\tilde{\Lambda}_i \|a_i - \hat{a}_i\|) + (\langle \hat{a}_i, x_0 \rangle - \hat{b}_i)^2 + O(\tilde{\Lambda}_i \|a_i - \hat{a}_i\|)$$
$$= O(\tilde{\Lambda}_i \|a_i - \hat{a}_i\| + v(a_i, x_0)).$$

Thus, we can now finish our bound on the second moment of X_t:

$$\mathbb{E}[X_t^2] = \frac{1}{s^2} \cdot \sum_{i=1}^{n} (\langle a_i, x \rangle - b_i)^4 \cdot \frac{\sum_{j \in [n]} \tilde{\Lambda}_i \|a_j - \hat{a}_j\| + v(a_j, x_0)}{\tilde{\Lambda}_i \|a_i - \hat{a}_i\| + v(a_i, x_0)}$$

$$\leq \frac{1}{s^2} \cdot \sum_{i=1}^{n} (\langle a_i, x \rangle - b_i)^2 \cdot O(\tilde{\Lambda}_i) \|a_i - \hat{a}_i\| + v(a_i, x_0)) \cdot \frac{+ \sum_{j \in [n]} \tilde{\Lambda}_j \|a_j - \mathcal{A}(a_j)\| + v(a_j)}{\tilde{\Lambda}_i \|a_i - \hat{a}_i\| + v(a_i, x_0)}$$

$$\leq \frac{\sum_{j \in [n]} \tilde{\Lambda}_j \|a_j - \mathcal{A}(a_j)\| + v(a_j, x_0)}{s^2} \cdot \sum_{i=1}^{n} O\left((\langle a_i, x \rangle - b_i)^2\right)$$

Using the same upper bounds, we get that for all i,

$$w(i)(\langle a_i, x \rangle - b_i)^2 = \frac{1}{s} \cdot (\langle a_i, x \rangle - b_i)^2 \cdot \frac{\sum_{j \in [n]} \tilde{\Lambda}_j \|a_j - \mathcal{A}(a_j)\| + v(a_j, x_0)}{\tilde{\Lambda}_i \|a_i - \hat{a}_i\| + v(a_i, x_0)}$$

$$\leq \frac{\sum_{j \in [n]} \tilde{\Lambda}_j \|a_j - \mathcal{A}(a_j)\| + v(a_j, x_0)}{s}.$$

Therefore, for $T = \varepsilon(\|Ax - b\|_2^2 + \Phi_\Lambda(A) + \sum_i v(a_i, x_0))$ we get that $2\sum_t \mathbb{E}[X_t^2] \leq T^2/s$, and with probability 1 each X_t is verifies $2|X_t|T/3 \leq T^2/s$. Furthermore, using Equation (2) and the optimality of x_0 for the dataset $\{\hat{a}_1, \ldots, \hat{a}_n\}$, we

18

have that:

$$\sum_i v(a_i, x_0) = \sum_i (\langle \hat{a}_i, x_0 \rangle - \hat{b}_i)^2 \leq \sum_i (\langle \hat{a}_i, x \rangle - \hat{b}_i)^2$$

$$\leq \sum_i (\langle a_i, x \rangle - b_i)^2 + O(\tilde{\Lambda}_i)\|a_i - \hat{a}_i\| = \|Ax - b\|_2^2 + O(\Phi_\Lambda(A)).$$

Hence, the Bernstein inequality ensures that

$$\Pr\left[\left|\sum_{e \in S} w(e)(\langle a_e, x \rangle - b_s)^2 - \|Ax - b\|_2^2\right| \geq \varepsilon(\|Ax - b\|_2^2 + \Phi_\Lambda(A)))\right]$$

$$\leq \Pr\left[\left|\sum_{e \in S} w(e)(\langle a_e, x \rangle - b_e)^2 - \|Ax - b\|_2^2\right| \geq \varepsilon/2 \cdot (\|Ax - b\|_2^2 + \Phi_\Lambda(A) + \sum_i v(a_i, x_0))\right]$$

$$\leq \exp\left(-\varepsilon^2 s/8\right).$$

Therefore, using that $s = 8\varepsilon^{-2}\log(1/\delta)$ concludes the lemma. □

C. More experimental details

C.1. Experimental Setup

For the clustering required in Algorithm 1, we run k''-means clustering using Python's sklearn implementation, for some $k'' < k$ on the model's last layer embeddings. Note that k is the total number of data points to be selected, while k'' is the number of cluster centers sampled (which are a subset of the points being selected). For our experiments, we chose $k'' = 0.2k$. Since the k'' cluster centers might not be actual data points from the dataset, we replace each center with the closest data point from the dataset in ℓ_2 norm (note that by triangle inequalities, this loses only a factor 4 in the k-means cost). After computing $\ell(e)$ for each center and extrapolating to the whole dataset using the approximation $\tilde{\ell}(e) := \ell(c_e) + \lambda \|e - c_e\|_2^2$ (where c_e is the closest center to e), we sample the remaining $k - k' - k''$ data points proportional to $\tilde{\ell}$.

C.2. Fine-Tuning LLMs

Model Details. For our fine-tuning task, we display the model hyperparameters in the table below. We used a batch size of 128, a constant learning rate of 0.001, and dropout of 0.1. We fine-tune from the T5-Small (Raffel et al., 2019) pre-trained model that was pre-trained for 1M steps.

Model Name	T5-Small
embedding dim	512
number of heads	6
enc./dec. layers	8
head dim	64
mlp dimension	1024
number of parameters	77M

Table 3. Dimensions of T5 Small

Hölder Constant. We experimented with varying the Hölder constant from 0.05 to 500. Each run used BERT embeddings for clustering (as above), and sampled roughly 1% of the whole dataset. We found that raising the constant too high (e.g., 500) resulted in a drop in quality and final accuracy more equivalent with diversity sampling (figure 5).

C.3. Classification Tasks

Datasets and models. We largely follow the dataset and model setup used in DISTIL. The datasets we consider are MNIST, Fashion MNIST and CIFAR10. For the first two we use a neural network with one 128-dimensional hidden layer, and for the last one we use convolutional neural network with three convolutional layers and three dense layers. We train each model for 10 epochs, a batch size of 32, and use the Adam optimizer with a learning rate of 10^{-3}.
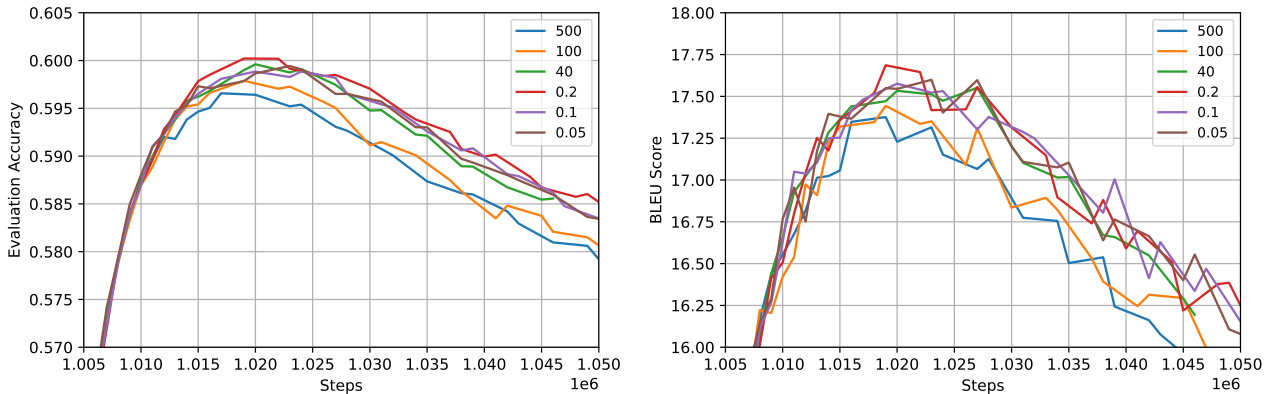
Figure 5. We report the accuracy (left) and BLEU score (right) with differing Hölder constants from 0.05 to 500. Each run uses roughly 1% of the whole dataset.
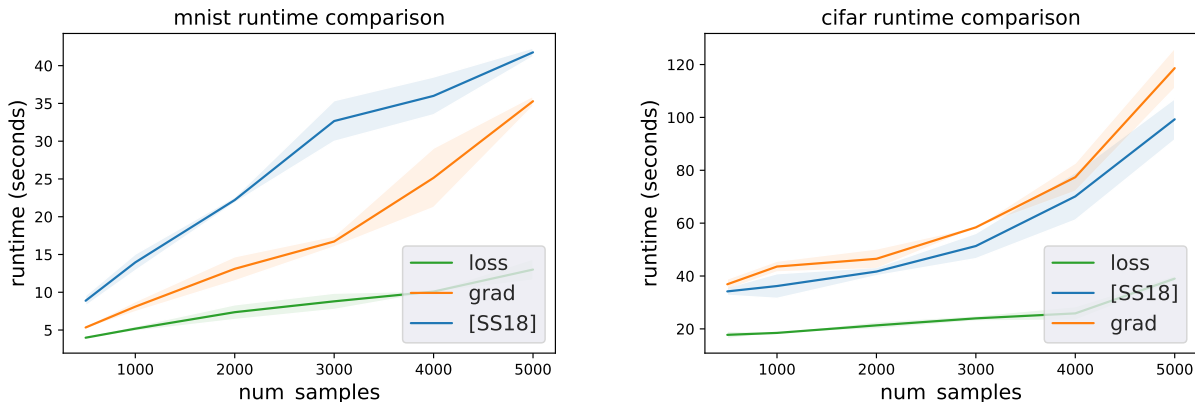

Figure 6. We present runtime comparisons between different algorithms, for MNIST and CIFAR10. The results for Fashion MNIST are analogous to those of MNIST, since the model and dataset have the same size.

Runtime comparison. In Figure 6, we show a comparison between the runtimes of our loss- and gradient-based algorithms, and the coreset algorithm of (Sener & Savarese, 2018). All algorithms were implemented in python using the tensorflow framework and the runtime calculation experiments ran on CPU, on a cloud VM with 24 CPUs and 100GB of RAM. It should be noted that a significant advantage of our loss- and gradient-based sampling is that they can rely on a pre-computed metric and clustering that is not updated during the sampling process. In most applications, this will be a fixed metric generated by an upstream model, that is easy to generate and compute distances. As a result, most of the runtime will be spent running model inferences at the cluster center points.

Algorithms for data selection. We list some of the best-performing algorithms in literature, and mention their advantages and disadvantages. Out of these, margin and entropy sampling are the top performing methods in the DISTIL[3] benchmark.

- Uniform sampling: We uniformly sample data points up to the budget k. This is the simplest and fastest way to sample k data points.

- Margin/Least confidence/Entropy sampling: These methods aim to select the examples with the lowest confidence. Specifically, if $p_1, \ldots, p_C$ are the per-class output probabilities of the model, we select the data points that either minimize $\max_{i \in [C]} p_i$, minimize $p_{i^*} - \max_{i \in [C] \setminus \{i^*\}} p_i$, where $i^* = \operatorname{argmax}_{i \in [C]} p_i$, or maximize the entropy $-\sum_{i=1}^{C} p_i \log p_i$. Unfortunately, these methods require an inference call for *each* data point, in order to evaluate its

---
[3]https://github.com/decile-team/distil

classification uncertainty, and so are not considered runtime-efficient.

- k-center CoreSet [SS18]: The k-center algorithm from (Sener & Savarese, 2018). This algorithm does not require any model inferences, but instead requires maintaining a nearest-neighbor data structure under insertions of data points.

In Figure 7 we provide more detailed experiments, including multiple algorithms from previous work.
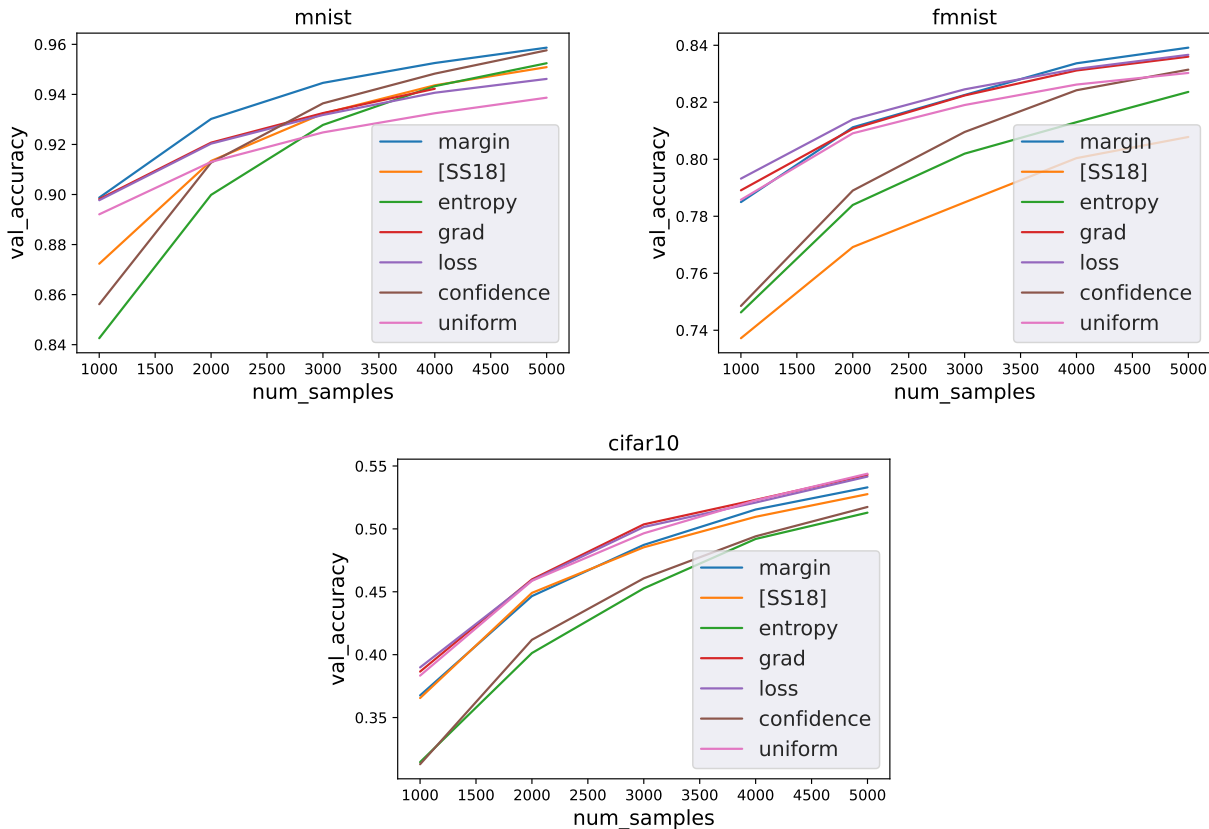

Figure 7. The experimental results for different datasets and algorithms. To minimize variation, we independently run each data point 100 times.

C.4. Regression tasks

Our algorithm. We run Algorithm 2, with a couple of differences: i) We run k-medoids, which is a variant of k-median that is only allowed to pick input points as centers – this does not change the theoretical guarantees provided in the previous sections, and ii) we set $\Lambda_i \to \infty$ for all i, which has the effect that we only look at distances and not losses. We set the number of clusters to be 10% of the total number of data points in the training set. After computing the regression solution, we evaluate it on the full training and validation datasets.

Runtime comparison on regression. In Table 4 we compare the runtimes of computing the leverage scores of a matrix and computing a k-medoid clustering (a variant of k-median). As the results show, even on matrices with i.i.d. normal entries, the standard k-medoid implementation from Python's scikit-learn library is significantly faster than computing the leverage scores. This is expected, since leverage score computation takes time $O(\mathrm{nnz} + n^\omega)$, where nnz is the number of non-zeros in the data matrix and $\omega \geq 2$ is the runtime exponent of matrix multiplication, while $O(1)$-approximate k-median takes time $O(\mathrm{nnz} + n)$.

n	5,000	10,000	20,000	30,000
Leverage runtime (sec)	36.62	250.64	1890.73	6098.31
k-medoid runtime (sec)	**19.74**	**105.29**	**858.5**	**2712.09**

Table 4. Comparing the runtime for computing all leverage scores and a k-medoids clustering for $k = 0.2n$, of an $n \times n$ matrix with i.i.d. standard normal entries.