# Tradeoffs for Space, Time, Data and Risk in Unsupervised Learning

**Mario Lucic**
ETH Zürich

**Mesrob I. Ohannessian**
University of California, San Diego

**Amin Karbasi**
Yale University

**Andreas Krause**
ETH Zürich

## Abstract

Faced with massive *data*, is it possible to trade off (statistical) *risk*, and (computational) *space* and *time*? This challenge lies at the heart of large-scale machine learning. Using $k$-means clustering as a prototypical unsupervised learning problem, we show how we can strategically summarize the data (control space) in order to trade off risk and time when data is generated by a probabilistic model. Our summarization is based on *coreset* constructions from computational geometry. We also develop an algorithm, TRAM, to navigate the space/time/data/risk tradeoff in practice. In particular, we show that for a fixed risk (or data size), as the data size increases (resp. risk increases) the running time of TRAM *decreases*. Our extensive experiments on real data sets demonstrate the existence and practical utility of such tradeoffs, not only for $k$-means but also for Gaussian Mixture Models.

## 1 INTRODUCTION

The computational and statistical performance of any learning algorithm for a given data set can be described in terms of three parameters: risk, running time, and space usage. The massive growth in datasets, coupled with limited resources in terms of time and space, raises new challenging questions on the accuracy of learning that can be achieved. At the heart of this challenge is to identify the relationships between *risk* $\varepsilon$, and the resources we have available, namely, *time* $t$, *space* $s$, and *data* $n$. Most of classical learning theory centers around the question of how risk scales with dataset (or sample) size: How much data $n$ is needed in order to achieve a certain level of risk $\varepsilon$ (i.e., what is the sample complexity of a given learning task)? In contrast, and from a practical point of view, increasing the data size is a source of computational complexity which typically translates into higher running time $t$. From this perspective, large data is considered a nuisance rather than a resource for achieving lower risk. As a result, most practical algorithms accumulate data until they exhaust either the time or space constraints and drop the data afterwards.

**Related Work.** An alternative direction is to investigate *computational and statistical tradeoffs*: using data as a computational resource when available beyond the sample complexity of the learning task. Pioneering this effort, Decatur et al. [2000] and Servedio [1999] showed tradeoffs in the realizable PAC learning model. Exploring these tradeoffs has gained much recent attention due to emerging problems in big data. For instance, Bottou and Bousquet [2008], Shalev-Shwartz and Srebro [2008] and Birnbaum and Shwartz [2012] showed the existence of such tradeoffs for learning linear classifiers as the data size increases. These tradeoffs are generally achieved by leveraging the fact that as we accumulate more data, the desired risk $\varepsilon$ becomes easier to reach, thus computationally cheaper but less accurate algorithms can be employed. This idea of *algorithmic weakening* was explored more systematically by Chandrasekaran and Jordan [2013] using convex relaxations.

**Our Contributions.** Existing approaches in computational and statistical tradeoffs consider only three of the four parameters: for a desired level of risk $\varepsilon$ they identify tradeoffs between running time $t$ and data size $n$. Our primary goal in this paper is to study how *summarization* (i.e., controlling space) can help navigate the tradeoff between time, data size and risk. In other words, we present a *weakening mechanism*, akin to Chandrasekaran and Jordan [2013], albeit in a different direction. Instead of weakening *learning algorithms*, we consider weakening the *data representation*. As more data becomes available, more representative elements can be extracted, without incurring much computational cost. Our approach is based on novel computational geometric techniques, called *coresets* (Agarwal et al. [2005]), where a small amount of most relevant data is extracted from the dataset, while performing the computation on this extracted data guarantees an approximate solution to the original problem. To the best of our knowledge, this paper is a first effort in introducing a methodological data-summarization approach for studying and navigating space/time/data/risk tradeoffs.

Appearing in Proceedings of the $18^{th}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38. Copyright 2015 by the authors.

As a prototypical unsupervised learning problem, we focus on $k$-means clustering, also known as *vector quantization*, due to its simplicity and practical importance. In this problem, a set of $k$ centers is sought to minimize the expected (squared) distance between data points and the closest center. Finding the optimal centers is NP-hard, but good approximation algorithms are known, e.g., Lloyd's algorithm (Lloyd [1982]). We show how coreset constructions for $k$-means (Kanungo et al. [2002], Har-Peled and Mazumdar [2004], Agarwal et al. [2005], Feldman et al. [2007, 2013]) can be used to strategically summarize the data: in order to achieve a fixed precision, the running time can be made to *decrease* as the data set grows, by carefully controlling space usage. We also provide a practical algorithm TRAM that uses existing algorithms for solving $k$-means (e.g., Lloyd's algorithm, or $k$-means++) in order to realize this tradeoff in practice. We demonstrate the effectiveness of our summarization strategy on several synthetic and real data sets. We should highlight that $k$-means clustering is a *non-convex* problem, thus prior computational-statistical tradeoff strategies that heavily relied on convexity cannot be applied in this setting. While we focus on $k$-means, coresets are available for many other unsupervised learning tasks (Feldman et al. [2013]), and we believe that our approach can be applied much more generally. In particular, we empirically demonstrate how such tradeoffs can be achieved for Gaussian Mixture Models (GMMs).

## 2 THE STATISTICAL $k$-MEANS PROBLEM

Typically, $k$-means is viewed as a (combinatorial) optimization problem. We focus instead on the statistical variant. In particular, we assume that an underlying distribution generates i.i.d. samples, and we seek centers that generalize well. More formally, let $\mathbf{P}$ be an *unknown* distribution on $\mathbb{R}^d$ where we assume that it is supported on a ball of radius $B$ at the origin, i.e., for $X \sim \mathbf{P}$ we have $\mathbf{P}(\|X\|_2 \leq B) = 1$ (this assumption can be relaxed under other regularity conditions, see, for example Telgarsky and Dasgupta [2013]). In $k$-means clustering, any data point $x \in \mathbb{R}^d$ is associated with the closest among a set of $k$ *centers* $c = \{c_1, \cdots, c_k\}$, where $c_i \in \mathbb{R}^d$. We judge the quality of this association by a *risk* defined as

$$R(c) = \mathbf{E}_{X\sim\mathbf{P}}[\mathrm{d}^2(c,X)]$$

between $c$ and a sample $X$ from $\mathbf{P}$, where $\mathrm{d}^2(c,X) = \min_{i=1}^k ||c_i - X||_2^2$. Let $\mathscr{C}$ be the set of all $k$ centers in the ball of radius $B$ at the origin. The *optimal centers* are those that minimize this risk:

$$c^\star = \arg\min_{c\in\mathscr{C}} R(c).$$

The solution to this minimization may not be unique, but for the ease of presentation we assume it is. We further make the realistic assumption that $\underline{R} := R(c^\star) > 0$ which is satisfied for any distribution supported on more than $k$ points. Since $\mathbf{P}$ is unknown, we seek centers for a dataset of $n$ samples $X_1, \ldots, X_n$ drawn i.i.d. from $\mathbf{P}$. Any choice of a sequence of functions $\tilde{c}_n$, from $\mathbb{R}^{d\times n} \to \mathbb{R}^{d\times k}$ is called a $k$-means *procedure*. Out of all such choices, of particular importance is the one that minimizes the *empirical risk*, to obtain the *empirically optimal centers*:

$$R_n(c) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{d}^2(c,X_i), \quad \hat{c}_n = \arg\min_{c\in\mathscr{C}} R_n(c). \tag{1}$$

The properties of the empirically optimal centers have been extensively studied in the literature ([Kanungo et al., 2002, Ben-David, 2007]). In particular, finding empirically optimal centers is a daunting task and often approximate procedures are used. Of particular interest to us is a class of algorithms (Kanungo et al. [2002], Har-Peled and Mazumdar [2004], Agarwal et al. [2005], Feldman et al. [2007, 2013]) that solve the $k$-means problem by first summarizing the data and then finding the centers on the summarized data. This decoupling principle allows these algorithms to invest most of their running time only on a small set of points and, at the same time, to save space.

## 3 DATA SUMMARIZATION

Data summarization refers to a procedure that takes a data set of size $n$ and replaces it with a smaller set of size $s_{\mathsf{proc}}$, which suffices for (approximately) solving the learning task at hand. This summarization may simply be a truncation without any consideration to the inherent structure of the data (a simple method that is often practiced), or it may be a combination of truncation and strategic sampling that adapts to structure in the data. We denote the truncation size by $m_{\mathsf{proc}}$. One of the main advantages of having summarized data, apart from saving space, is the substantial reduction in running time. For this reason, truncation must be allowed, as otherwise the running time of *any* learning algorithm would grow with the data size. We now formally present these two strategies.

**Uniform Subsampling** This is the simplest form of data summarization: start with a data set of size $n$, preserve only the first $s_{\mathsf{subs}} \leq n$ points, and then solve the learning problem by minimizing the empirical risk. In the $k$-means problem, this amounts to $\tilde{c}_{\mathsf{subs}} = \arg\min_{c\in\mathscr{C}} R_{s_{\mathsf{subs}}}(c)$ where $R_{s_{\mathsf{subs}}}(c) = \frac{1}{s_{\mathsf{subs}}}\sum_{i=1}^{s_{\mathsf{subs}}} \mathrm{d}^2(c,X_i)$. For the uniform subsampler the summarization and truncation sizes are identical, $s_{\mathsf{subs}} = m_{\mathsf{subs}}$. Larger values of $s_{\mathsf{subs}}$ promote lower statistical risk but are more expensive to compute. Conversely, computation on a smaller set may be fast but results in higher risk. The uniform subsampler may tune $s_{\mathsf{subs}}$ to balance risk with running time.

**Strategic Sampling** *Coresets* are data summaries that are constructed via adaptive sampling, in the spirit of importance sampling. As with the uniform subsampler, we start with data of size $n$, then truncate it to $m_{\mathsf{core}}$ points. Now, instead of using the truncation as is, we perform strategic sampling to propose a set of $s_{\mathsf{core}}$ representative points $(Y_j)_{j=1,\cdots,s_{\mathsf{core}}}$, each associated with a non-negative weight $w_j$, and we solve the learning problem not on the empirical risk, but on a *weighted* variant. In the $k$-means problem, this amounts to $\tilde{c}_{\mathsf{core}} = \arg\min_{c\in\mathscr{C}} R^w_{s_{\mathsf{core}}}(c)$ where $R^w_{s_{\mathsf{core}}}(c) = \sum_{j=1}^{s_{\mathsf{core}}} w_j \mathrm{d}^2(c, Y_j)$. Coresets strive to be a more faithful/concise representation of the data than uniform samples. Naturally, their properties depend on how the strategic sampling is performed. The hallmark property of coresets is their ability to approximate the empirical risk, defined in (1), optimized over the starting $m_{\mathsf{core}}$ data points.

**Definition 1.** *A coreset construction is a $(1+\eta)$-approximation, with $\eta$ a function of the coreset size $s_{\mathsf{core}}$, if the centers $\tilde{c}_{\mathsf{core}}$ satisfy $R_{m_{\mathsf{core}}}(\tilde{c}_{\mathsf{core}}) \leq (1+\eta(s_{\mathsf{core}}))R_{m_{\mathsf{core}}}(\hat{c}_{m_{\mathsf{core}}})$.* [1]

A coreset procedure could start out with a moderately larger truncation $m_{\mathsf{core}} > m_{\mathsf{subs}}$, and yet produce a representation that is significantly smaller $s_{\mathsf{core}} \ll s_{\mathsf{subs}}$, all while maintaining a comparable risk. Note again that without performing truncation, the running time of finding a coreset of size $s_{\mathsf{core}}$ using the whole dataset grows with the data size. A number of efficient $(1+\eta)$-coreset constructions for $k$-means are known, as reviewed in Section 5.2. We study a particularly practical variant in Section 7. Additionally, it is worth noting that coresets have the advantage of admitting *streaming* and *parallel* constructions (Har-Peled and Mazumdar [2004], Balcan et al. [2013]), which makes them particularly suited for massive datasets.

## 4 SPACE-TIME-DATA-RISK TRADEOFF

Our goal now is to give a precise definition of tradeoffs: how data summarization may lead to trading off representation space, running time, data size, and statistical risk. Let $\tilde{c}_{\mathsf{proc}}(n, m_{\mathsf{proc}}, s_{\mathsf{proc}})$, or $\tilde{c}_{\mathsf{proc}}$ for short, denote a $k$-means procedure based on data summarization, such as uniform subsampling or coreset summarization. Recall that such a procedure starts with $n$ data points, truncates them to $m_{\mathsf{proc}}$ points, summarizes these to $s_{\mathsf{proc}}$ (possibly weighted) representative points, and optimizes the (possibly weighted) empirical risk to obtain the set of centers $\tilde{c}_{\mathsf{proc}}$. The *running time*, which we denote by $t_{\mathsf{proc}}$, may be further decomposed into: summarization time $t^{\mathsf{sum}}_{\mathsf{proc}}$ and the time $t_{\mathsf{solver}}$ for empirical risk optimization. The former depends on the particular procedure, but the latter can be a generic solver across procedures. We assume that the act of truncation (for both the uniform subsampler and the coreset procedure) has no computational cost. The *statistical risk* of the procedure, which we denote by $R_{\mathsf{proc}}$, is the expected risk, where the expectation is taken with respect to the sample. That is, $R_{\mathsf{proc}} := \mathbf{E}[R(\tilde{c}_{\mathsf{proc}})]$. We can decompose it as follows:

$$\begin{aligned} R_{\mathsf{proc}} \quad &\leq \quad \underbrace{R(c^\star)}_{\varepsilon_{\mathsf{model}}} + \underbrace{\mathbf{E}[R(\hat{c}_{m_{\mathsf{proc}}})] - R(c^\star)}_{\varepsilon_{\mathsf{est}}} \\ &+ \quad \underbrace{|\mathbf{E}[R(\tilde{c}_{\mathsf{proc}})] - \mathbf{E}[R(\hat{c}_{m_{\mathsf{proc}}})]|}_{\varepsilon_{\mathsf{sum}}}, \end{aligned} \tag{2}$$

where $\varepsilon_{\mathsf{model}}$, $\varepsilon_{\mathsf{est}}$, and $\varepsilon_{\mathsf{sum}}$ are the *modeling*, *estimation*, and *summarization* errors, respectively. The modeling error is the best risk achieved by any $k$ centers (limitation of the model). The estimation error is incurred due to using the *empirically* optimal centers (limitation of estimating from data). Lastly, we have the error of *approximate* data summarization. For coresets it depends on $\eta$ (cf. Proposition 4).

**How to trade off** The four dimensions *space, time, data, and risk* put forth in this paper can now be represented by the four parameters $(s_{\mathsf{proc}}, t_{\mathsf{proc}}, m_{\mathsf{proc}}, R_{\mathsf{proc}})$. We can obtain a variety of tradeoffs by constraining some dimensions and optimizing others. Of course, not all $(s,t,m,R)$-tuples are attainable: for instance, classical sample complexity bounds constrain what risks are attainable at what data sizes. We call a subset of the dimensions *feasible* for a procedure, if there exist values of the others that lead to attainable tuples. By exploring the feasible landscape, one can harness various trends. For example, based on the risk decomposition stated above, as we decrease $s_{\mathsf{proc}}$, the risk $R_{\mathsf{proc}}$ increases due to the increase in $\varepsilon_{\mathsf{sum}}$. In contrast, solving the optimization becomes computationally cheaper with smaller $s_{\mathsf{proc}}$. These interactions, illustrated schematically in Figure 1b give rise to various tradeoffs. Some of these are listed in Figure 1a.

In this paper, we are mainly interested in *(a) data-time tradeoffs*: for $R_{\mathsf{proc}}$ fixed below some $\varepsilon_{\mathsf{total}}$, can $t_{\mathsf{proc}}$ decrease as $n$ increases? and *(b) risk-time tradeoffs*: for some fixed $n$, can $t_{\mathsf{proc}}$ decrease as $R_{\mathsf{proc}}$ increases? These two tradeoffs are listed respectively in the first and second rows of the table in Figure 1a. Data summarization gives us a natural framework to answer those questions: we could achieve such gains by optimizing summarization space $s_{\mathsf{proc}}$. This captures the weakening-through-data-summarization mechanism that we advocate in this paper. Formally, given a data size $n$ and risk $\varepsilon_{\mathsf{total}}$, the *optimal running time* function is:

$$\begin{gathered} t^\star_{\mathsf{proc}}(n, \varepsilon_{\mathsf{total}}) = \min_{m_{\mathsf{proc}}, s_{\mathsf{proc}}} t_{\mathsf{proc}}(n, m_{\mathsf{proc}}, s_{\mathsf{proc}}), \\ \text{s.t. } R_{\mathsf{proc}}(m_{\mathsf{proc}}, s_{\mathsf{proc}}) \leq \varepsilon_{\mathsf{total}}, m_{\mathsf{proc}} \leq n. \end{gathered} \tag{3}$$

Observe that for fixed $\varepsilon_{\mathsf{total}}$ and as $n$ varies, the optimal running time $t^\star_{\mathsf{proc}}$ is non-increasing in $n$ by construction. Similarly, for fixed $n$ and as $\varepsilon_{\mathsf{total}}$ varies, the optimal running time $t^\star_{\mathsf{proc}}$ is non-increasing in $\varepsilon_{\mathsf{total}}$.

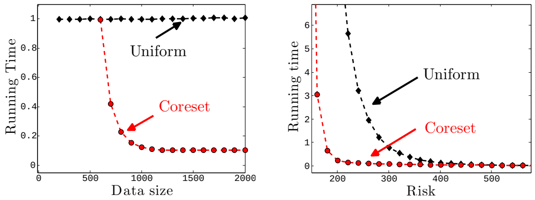
[1] Coresets conventionally require approximating the risk at *all* $c$: for $\varepsilon \in (0,1)$, $\forall c \in \mathscr{C}$, $|R^w_{s_{\mathsf{core}}}(c)/R_m(c) - 1| \leq \varepsilon$. This implies a $(1+\eta)$-approximation with $\eta = 2\varepsilon/(1-\varepsilon)$.

| Tradeoff | Space | Time | Data | Risk |
|---|---|---|---|---|
| Data-Time | Tune | Objective | Vary | Fixed |
| Risk-Time | Tune | Objective | Fixed | Vary |
| Space-Risk | Vary | Tune | Fixed | Objective |
| Data-Risk | Tune | Fixed | Vary | Objective |
| Space-Time | Vary | Objective | Tune | Fixed |

(a) Tradeoffs

| Cost | k↗ | n↗ | s↗ |
|---|---|---|---|
| $\varepsilon_{\text{modelling}}$ | ↘ | | |
| $\varepsilon_{\text{estimation}}$ | ↗ | ↘ | |
| $\varepsilon_{\text{summarization}}$ | ↗ | → | ↘ |
| $t$ | ↗ | ↗ | ↗ |

(b) Trends



(c) Data-Time

(d) Risk-Time

Figure 1: (a) Examples of Space-Time-Data-Risk-Tradeoffs, each realized by trading off two parameters (green and gray), by constraining (red) and tuning (blue) the remaining ones. (b) Effect of increasing $k$, $n$ and $s$ on the various errors and running time $t$. (c) Coreset (red) data-time tradeoffs versus subsampler (black). The plots represent best running time for fixed risk tolerance when varying the data size, as predicted by our theory (Section 5). (d) Risk-time tradeoff, i.e., best achievable running time for fixed data size when varying the allowed risk. [Time units normalized to the median subsampler time.]

**Definition 2.** *We say that a k-means procedure offers a (non-trivial)* data-time tradeoff *if, for a given desired total risk $\varepsilon_{\text{total}}$, the running time $t^\star_{\text{proc}}(\cdot, \varepsilon_{\text{total}})$ is decreasing for some range of $n$. We say that the procedure offers a (non-trivial)* risk-time tradeoff *if, for a given data size $n$, $t^\star_{\text{proc}}(n, \cdot)$ is decreasing for some range of $\varepsilon_{\text{total}}$. In other words, these tradeoffs correspond to (non-flat)* Pareto optimal frontiers *of $t^\star_{\text{proc}}$, as either of the arguments is fixed.*

Tradeoffs divide the landscape into various operation regimes. For data-time tradeoffs, before $n$ reaches the feasible range for $\varepsilon_{\text{total}}$, we are in a "data-bounded" regime (cf., Shalev-Shwartz and Srebro [2008]). We cannot get the desired risk $\varepsilon_{\text{total}}$, and have to invest all of the data and computation to driving the risk as low as possible. On the other extreme, very large data sizes are bound to lead to a point where more data can safely be discarded with no further impact on risk and computation time. This is the "data-laden" regime. In our framework, it means that in the data-laden regime $t^\star_{\text{proc}}(\cdot, \varepsilon_{\text{total}})$ flattens. Lastly, there is an "intermediate regime" where all of the available data is used, but there is maneuvering room to drive the computation time down or in other words $t^\star_{\text{proc}}(\cdot, \varepsilon_{\text{total}})$ decreases. A lot of the subtlety of the tradeoffs happens in this regime. We see this phenomenon manifest itself both analytically, in Section 5, and experimentally, in Section 7.

**Extensions** Our methodology is formalized for the $k$-means problem, but the framework is much richer. For example, spectral clustering methods that can be mapped to $k$-means are bound to profit directly from our results. A concrete extension consists of Gaussian Mixture Models, by using the negative log-likelihood as the risk and coreset construction by Feldman et al. [2011]. We do not formalize this, but we demonstrate it experimentally in Section 7.

# 5 ANALYSIS

We have thus far motivated and laid out a clear paradigm of tradeoffs via data summarization. But are such tradeoffs even possible? In this section, we show that the answer is *yes*. To keep our exposition concise, we focus in particular on showing that nontrivial data-time tradeoffs (Definition 2) do indeed exist. For this we need to characterize $t^\star_{\text{proc}}(n, \varepsilon_{\text{total}})$ as $n$ varies, for a fixed desired risk level $\varepsilon_{\text{total}}$.

For the uniform subsampler the data-time tradeoff is necessarily trivial. To see this, let $n_{\text{f}}(\varepsilon_{\text{total}})$ be the smallest data size $n$ when $\varepsilon_{\text{total}}$ becomes feasible. Then for all $n \geq n_{\text{f}}$ we have $\varepsilon_{\text{model}} + \varepsilon_{\text{est}}(n) \leq \varepsilon_{\text{total}}$ but the uniform subsampler has no incentive to use more than $m_{\text{subs}} = n_{\text{f}}$ samples, since otherwise its running time would be greater (for unneeded risk reduction). This means that $t^\star_{\text{subs}}(\cdot, \varepsilon_{\text{total}})$ is undefined for $n < n_{\text{f}}(\varepsilon_{\text{total}})$, and is flat beyond that. In the language of Section 4, the uniform subsampler switches abruptly from the "data-bounded" to the "data-laden" regime.

The more interesting question is thus: Can coreset procedures give non-trivial data-time tradeoffs that improve on the uniform subsampler? In particular, can we observe an "intermediate regime" where $t^\star_{\text{core}}(\cdot, \varepsilon_{\text{total}})$ curves down, before reaching the data-laden regime? Our main result answers these questions in the affirmative. Informally, we have the following.

**Main Result** (Existence of Tradeoffs)**.** *Let the following conditions hold for a coreset procedure:*

*(a) The summarization is time-efficient (its running time is negligible relative to that of the solver).*

*(b) The summarization is sample-efficient (the approximation factor vs. summarization size decays faster than the estimation error vs. sample size).*

*(c) The estimation error decays fast ($\sim$ power law).*

*(d) The solver is slow (at least super-linear).*

*Then, for small enough risks, the procedure admits a non-trivial data-time tradeoff, and its optimal running time dominates (is less than) that of the uniform subsampler for large enough data sizes. Moreover, existing bounds and coreset constructions do satisfy these conditions.*

In what follows, we proceed to formalize this result. In Section 5.1 we give the sufficient conditions and in Section 5.2 we affirm that these conditions are satisfied in practice, by giving existing risk bounds and coreset constructions. We also provide some numerical illustrations of tradeoffs using these bounds. In Section 7 we demonstrate these tradeoffs experimentally.

### 5.1 Sufficient Conditions for Tradeoffs

Recall first some notation from Section 4. When a coreset summarization procedure has a total risk ($R_{\mathsf{core}}$), it can be decomposed into modeling ($\varepsilon_{\mathsf{model}}$), estimation ($\varepsilon_{\mathsf{est}}$), and summarization errors ($\varepsilon_{\mathsf{sum}}$). The latter depends on the coreset approximation that results from a choice of a given summarization size ($\eta(s_{\mathsf{core}})$) (Proposition 4 makes this precise). The total running time of the procedure $t_{\mathsf{proc}}$ can be decomposed into summarization time ($t^{\mathsf{sum}}_{\mathsf{proc}}$) and empirical risk minimization time ($t_{\mathsf{solver}}$). The latter is attributed to a generic solver, and it depends only on the size ($s_{\mathsf{proc}}$) of its input. For the former, we add some further notation due to "bicriteria"-type coreset constructions (Feldman and Langberg [2011]), where the summarization stage itself is decoupled into two: initialization, taking time $t^{\mathsf{init}}_{\mathsf{core}}(m_{\mathsf{core}})$ that depends only on the (truncated) data size, followed by adaptive sampling, with time $t^{\mathsf{samp}}_{\mathsf{core}}(s_{\mathsf{core}})$ that depends only on the coreset summarization size. We are now ready to formally state our main result's conditions.

**Theorem 1.** *Let $t_{\mathsf{solver}}(\cdot)$, $t^{\mathsf{init}}_{\mathsf{core}}(\cdot)$, $t^{\mathsf{samp}}_{\mathsf{core}}(\cdot)$ be increasing, and $\varepsilon_{\mathsf{est}}(\cdot)$ and $\eta(\cdot)$ be decreasing functions of their arguments. Let the setting of the coreset procedure be such that the following are satisfied:*

*(a)* $t^{\mathsf{init}}_{\mathsf{core}}(\cdot)$ *is linear and* $t^{\mathsf{samp}}_{\mathsf{core}}(x) = o(t_{\mathsf{solver}}(x))$,

*(b)* $\exists a, b > 0$ *such that for large enough* $x$, $2\eta(x) \leq (1/\varepsilon_{\mathsf{model}} - a)\,\varepsilon_{\mathsf{est}}((1+b)x)$.

*(c)* $\forall L(x) \to \infty$, *no matter how slowly,* $\frac{\varepsilon_{\mathsf{est}}(xL(x))}{\varepsilon_{\mathsf{est}}(x)} \to 0$, *as* $x \to \infty$,

*(d)* $t_{\mathsf{solver}}(\cdot)$ *is bounded from below by a convex superlinear function, i.e.* $\frac{t_{\mathsf{solver}}(x)}{x} \to \infty$, *as* $x \to \infty$,

*Then there exists a small enough risk $\varepsilon_0$, such that for all desired risks $\varepsilon_{\mathsf{total}} \leq \varepsilon_0$, there exists a large enough sample size $n_0$, beyond which for all $n > n_0$ we have $t^{\star}_{\mathsf{core}}(n, \varepsilon_{\mathsf{total}}) < t^{\star}_{\mathsf{subs}}(n, \varepsilon_{\mathsf{total}})$.*

Since the coreset procedure cannot be faster than the subsampler at a sample size at the threshold of feasibility, the theorem implies that for all $\varepsilon_{\mathsf{total}} \leq \varepsilon_0$ the coreset procedure achieves a non-trivial tradeoff with an "intermediate regime", eventually dominating the uniform subsampler for large enough sample sizes.

Condition (a) asks for the solver's running time to overshadow that of summarization (how could one benefit from summarization otherwise?). Slower solvers can only "help" satisfy this condition. Condition (b) is more subtle, though it can be understood as follows: if larger summaries do not drive the summarization error down as fast as larger sample sizes drive the estimation error down, then summarization loses its competitive advantage against truncation. As for Conditions (c) and (d), they are primarily used in a technical context, to balance asymptotic expressions. As we outline in Section 5.2, these conditions are natural behaviors for the estimation error and solver respectively.

*Proof sketch of Theorem 1.* To prove this theorem, it suffices to show that for a large enough sample size $x$ we can find a (possibly suboptimal) coreset size $s$ such that the resulting procedure has $\varepsilon_{\mathsf{total}} = \varepsilon_{\mathsf{model}} + \varepsilon_{\mathsf{est}}(x)$ while its running time is less than $t_{\mathsf{solver}}(x)$. This is because $x$ and $t_{\mathsf{solver}}(x)$ represent respectively the feasibility threshold and the optimal running time of the uniform subsampler that achieves a risk of $\varepsilon_{\mathsf{total}}$. To maintain a risk of $\varepsilon_{\mathsf{total}}$, the coreset procedure needs an appropriate truncation size $m$ slightly larger than $x$, thus allowing enough samples for summarization, and the result would only hold for $n \geq m$.

We make a simple choice, $s = t^{-1}_{\mathsf{solver}}((1-2\delta)t_{\mathsf{solver}}(x))$ for some $\delta > 0$, ignoring rounding. This implies a performance gap $t_{\mathsf{solver}}(x) - t_{\mathsf{solver}}(s)$ of $2\delta t_{\mathsf{solver}}(x)$ within which we can maneuver. Then Condition (a) implies that for large enough $x$ the sampling stage will occupy less than $\delta t(x)$ of this gap. On the other hand, the initialization stage depends linearly on the resulting $m$. Condition (b) then intervenes to show that the impact of this stage remains also within another $\delta t(x)$, thus establishing the theorem. This, however, requires $x$ to be large enough to align with the constants of Condition (b), and for that we invoke Conditions (c) and (d). The details can be found in the supplements. □

### 5.2 Existence of Tradeoffs

We now affirm that the conditions of Theorem 1 are met by existing constructions.

**Proposition 1.** *Under known risk bounds (Propositions 2 and 3) and coreset constructions (Feldman and Langberg [2011]), and when using a super-linear polynomial-time or slower solver, the conditions of Theorem 1 are satisfied.*

We can illustrate this result visually via simulations: we perform numerical optimization using the risk, running time, and summarization bounds given in this section. The details can be found in the supplements. We plot a representative data-time tradeoff of both the subsampler (in black) and the coreset procedure (in red) in Figure 1c. Note that the coreset procedure dominates. The same type of numerical optimization can be done to obtain other tradeoffs: we plot the risk-time tradeoff of the same problem in Figure 1d. As Theorem 1 predicts, the coreset dominates primarily for smaller (thus more interesting) values of the risk. The proof of Proposition 1, also in the supplments, is a direct verification of the conditions of Theorem 1. We give here an account of the invoked bounds and coreset construction.

**Risk Bounds** The following bounds characterize the risks in terms of the parameters of the problem: the dimension $d$, radius $B$, and number of clusters $k$. Note that the modeling error does not depend on the procedure, the estimation error only depends on the procedure through the truncation size $m_{\mathsf{proc}}$, and the summarization errors depend more closely on the specifics of the summarization. The following bound on the modeling error is minimax up to constants (Graf and Luschgy [2000]).

**Proposition 2** (Modeling Error)**.** *The modeling error satisfies* $\varepsilon_{\mathsf{model}} \leq \frac{B^2 d}{k^{2/d}}$.

The estimation error has been extensively studied in statistics. We have the following (Antos et al. [2005]):

**Proposition 3** (Estimation Error)**.** *The estimation error satisfies* $\varepsilon_{\mathsf{est}} \leq \overline{\sigma} B^2 \frac{\sqrt{kd}}{\sqrt{m_{\mathsf{proc}}}}$, *for some* $\overline{\sigma} > 0$. *Furthermore, we have a lower bound: there exists* $\underline{\sigma} > 0$ *such that whenever* $k \geq 3$, *we may find* $\mathbf{P}$ *for which for large enough* $m_{\mathsf{proc}}$ *we have:* $\varepsilon_{\mathsf{est}} \geq \underline{\sigma} B^2 \frac{\sqrt{k^{1-4/d}}}{\sqrt{m_{\mathsf{proc}}}}$.

The summarization error depends on the particular summarization procedure. For uniform subsampling, since $\tilde{c}_{\mathsf{subs}} = \hat{c}_{m_{\mathsf{subs}}}$, it is trivially zero (cf. Equation (2)). For a coreset procedure, it depends on the coreset size or equivalently the approximation factor $\eta$.

**Proposition 4** (Summarization Error)**.** *Given a* $(1+\eta)$*-approximation coreset, when* $\eta(s_{\mathsf{core}}) \geq \eta_0 > 0$, *then* $\varepsilon_{\mathsf{sum}} < 2(\varepsilon_{\mathsf{model}} + \varepsilon_{\mathsf{est}})\eta(s_{\mathsf{core}})$ *for large enough m.*

Propositions 2 and 3 are restatements. On the other hand, Proposition 4 is new. The proof relies on uniform concentration (Linder [2002]), and is detailed in the supplements.

**Running Time Bounds** Solving for the exact empirically optimal centers is NP-hard, with the running time of known exact algorithms being $t_{\mathsf{solver}}(s) = \Omega(s^{kd})$ (cf., Inaba et al. [1994]). There are various popular heuristics, including Lloyd's ("the $k$-means") algorithm, and on typical inputs these have polynomial running times $t_{\mathsf{solver}}(s) = \Omega(\mathrm{poly}(k)\mathrm{poly}(d)\mathrm{poly}(s))$. Under further conditions they can be exact (Meyerson et al. [2004]). Even these optimistic polynomial running times are sufficient for us.

The uniform subsampler performs no summarization beyond truncation, $s_{\mathsf{subs}} = m_{\mathsf{subs}}$. Thus $t^{\mathsf{sum}}_{\mathsf{subs}} = 0$, and:

$$t_{\mathsf{subs}}(n, m_{\mathsf{subs}}, s_{\mathsf{subs}}) = t_{\mathsf{solver}}(s_{\mathsf{subs}}).$$

For coresets, we use the above-mentioned "bicriteria" construction by Feldman and Langberg [2011]. We have:

$$t_{\mathsf{core}}(n, m_{\mathsf{core}}, s_{\mathsf{core}}) = \\ t_{\mathsf{solver}}(s_{\mathsf{core}}) + t^{\mathsf{init}}_{\mathsf{core}}(m_{\mathsf{core}}) + t^{\mathsf{samp}}_{\mathsf{core}}(s_{\mathsf{core}}). \quad (4)$$

Like the risks, these initialization and sampling times depend on the various parameters of the problem, and in particular the dimension $d$ and the number of clusters $k$. In many constructions, these are *linear* functions of their arguments. In particular, the coreset construction of Feldman and Langberg [2011] is a $(1+\eta)$-approximation with $t^{\mathsf{init}}_{\mathsf{core}}(m_{\mathsf{core}}) = O(dkm_{\mathsf{core}})$ and $t^{\mathsf{samp}}_{\mathsf{core}}(s_{\mathsf{core}}) = O(s_{\mathsf{core}})$.

**Coreset Approximation** The last component of Proposition 1, needed to fully characterize a coreset approximation, is the functional relationship between the approximation factor $\eta$ and the coreset size $s_{\mathsf{core}}$. In particular, we note that Feldman and Langberg [2011] gives a $(1+\eta)$-approximation with a coreset of size $s_{\mathsf{core}} = O(dk(2+\eta)^2/\eta^2)$ (see also footnote[1]). We may thus write $\eta(s_{\mathsf{core}}) = O\left(\sqrt{dk}/(\sqrt{s_{\mathsf{core}}} - \sqrt{dk})\right)$.

## 6 DATA-DRIVEN TRADEOFF NAVIGATION

So far we demonstrated tradeoffs in $k$-means by considering analytical models. In practice, however, even if a tradeoff exists, it is a priori unclear how to harness it: one would seemingly need a "tuning oracle" to adjust the procedure to yield an optimal tradeoff, by selecting optimal truncation and summarization sizes. An exhaustive search for such an adjustment is useful for illustration, but it defeats the purpose of the endeavor, which is to yield a practical algorithm whose running time decreases with more data. In this section, we address this challenge by proposing a *TRadeoff nAvigation algorithM (*TRAM*)*. It uses a limited amount of additional validation data to explore the summarization landscape, and leads to a summarization that exhibits acceptable loss in risk $\varepsilon_{\mathsf{total}}$, time $t^\star$, and space $s^\star$, thus effectively approximating a tuning oracle. We focus specifically on data-time tradeoffs via coreset data-summarization schemes, though the approach is potentially extensible to other tradeoffs and procedures.

**Theoretical Setting** We design and study our algorithm under the following assumptions.

(A) The running time of the coreset procedure is known, up to scaling. In particular, we use a polynomial time solver and take $t_{\mathsf{core}} = \alpha m + s^\beta$ for known $\alpha, \beta > 1$.

(B) Evaluating the empirical risk using a data set of size $a$ takes a running time of $ka$.

(C) Let $m^\star$ and $s^\star$ be the solutions of Equation (3) realizing the optimal time $t^\star = t^\star_{\mathsf{core}}(n, \varepsilon_{\mathsf{total}})$. We have $R(\tilde{c}(m,s)) \leq \varepsilon_{\mathsf{total}}$ for all $m \geq m^\star$, $s \geq s^\star$, with probability at least $1-\lambda$.

(D) We have access to additional samples from the distribution $\mathbf{P}$, beyond the data size $n$.

Assumption (A) maps to the framework of Section 5.2: $t^{\mathsf{init}}_{\mathsf{core}}(m)$ is linear in $m$, $t^{\mathsf{samp}}_{\mathsf{core}}$ is absorbed into $t_{\mathsf{solver}}$, $t_{\mathsf{solver}}$ is polynomial, and both are normalized to maintain only a single constant. Assumption (B) is trivial, except for absorbing the dimension and leading constants into $k$. (C) is a monotonicity assumption, requiring that with some probability $1-\lambda$ not just the optimal coreset size *but also all larger* summaries are below the base risk $\varepsilon_{\mathsf{total}}$. The algorithm does not use $\lambda$, it is there only for performance analysis. Lastly, Assumption (D) uses separate data to validate in order to both use independence from the data itself, and allow to derive sample complexities for validation using basic concentration inequalities. Theorem 2 shows that only a small number of such points are needed. In practice, the data itself is partitioned to provide these points.

**A TRadeoff nAvigation algorithM (TRAM)** The idea of TRAM is as follows: search for a good summarization by starting small then growing until the desired risk is achieved. The challenge is that the risk cannot be known exactly and needs to be tested using data. We therefore have a compromise: if we stop too early we miss the target, and if we stop too late we spend too much on computation. The analysis shows that the algorithm achieves a certain balance.

**Algorithm** TRadeoff nAvigation algorithM (TRAM)

```
1: Input: Data of size $n$; risk level $\varepsilon_{\text{total}}$; validation data of size $a$; accuracy parameter $\delta > 0$.
2: Initialization: Start with a truncation of size $m[0] < n$ and a coreset size of $s[0]$.
3: repeat
4:     Iteration step $i$: Summarize the $m[i]$-truncation to a coreset of size $s[i]$, and solve for the centers $\tilde{c}[i]$. Increment $m[i]$ to $m[i+1]$, and $s[i]$ to $s[i+1]$. Use a portion $a[i]$ of the validation data to evaluate the empirical risk of $\tilde{c}[i]$.
5: until $R_{a[i]}(\tilde{c}[i]) \le \tau$.
6: Output: The last set of centers $\tilde{c}[i]$.
```

The validation data is a growing sequence drawn from the points described in Assumption (D). More specifically, $4b\log(1/\delta)/{\varepsilon_{\text{total}}}^2$ additional points are used at each iteration, where $b = 2B^2$, and thus $a[i] = 4ib\log(1/\delta)/{\varepsilon_{\text{total}}}^2$. The size increments happen multiplicatively: $m[i+1] \leftarrow \gamma_m m[i] \wedge n$ and $s[i+1] \leftarrow \gamma_s s[i]$. In particular, we take $\gamma_m = 2$ and $\gamma_s = 2^{1/\beta}$. Lastly, the threshold (in step 5) is $\tau = 3\varepsilon_{\text{total}}/2$.

**Theorem 2.** *Let $T$ and $J$ denote the running time and number of iterations of* TRAM *respectively. Under assumptions (A) to (D), given data of size $n$, a base risk $\varepsilon_{\text{total}}$, and parameter $\delta < \frac{1}{5}$, with probability at least $(1-\lambda)(1-5\delta)$,* TRAM*:*

▷ *runs for time $T \le 4t^{\star 2} + \frac{8bk}{{\varepsilon_{\text{total}}}^2}\log\frac{1}{\delta}\log_2^2 t^\star$,*

▷ *uses $a[J] \le \frac{8b}{{\varepsilon_{\text{total}}}^2}\log\frac{1}{\delta}\log_2 t^\star$ validation points,*

▷ *and produces centers $\tilde{c}$ with risk $R(\tilde{c}) \le 2\varepsilon_{\text{total}}$.*

*Proof sketch.* Using the validation test at every step, growing the set to compensate for dependencies, we control the errors of stopping too far before and too far after the optimal truncation and coreset sizes. The threshold that is slightly larger than the base risk gives us a detection margin. If these errors are not too large, the polynomial structure of the running time of the coreset procedure compounds with the geometric incrementing scheme, to lead to a computational overhead that remains reasonably close to the optimal. □

Note that the search does come with a penalty (the running time is squared). However, the analysis is very conservative and none of the constants depend on the data size $n$. Thus TRAM does indeed reproduce the qualitative behavior of the tradeoff, i.e. the running time decays as the data size increases, while the guaranteed risk remains effectively constant.

## 7 EXPERIMENTAL RESULTS

We now empirically establish the existence of tradeoffs and evaluate the performance of TRAM.

**Setup** Given a dataset $\mathscr{X} \subseteq \mathbb{R}^d$ and some $\varepsilon_{\text{total}}$, we wish to find the minimum computational cost of obtaining a $k$-means solution with risk less than or equal to $\varepsilon_{\text{total}}$. We interpret $\mathbf{P}$ as the uniform distribution over $\mathscr{X}$, hence we can compute the risk exactly. We simulate various dataset sizes by restricting individual experiments to a random subset of $\mathscr{X}$. For each pair of data size $n_i \in \mathscr{N}$ and summary size $s_j \in \mathscr{S}$ we sample $n_i$ instances i.i.d. from $\mathscr{X}$ and summarize the sample with a summary of size $s_j$ and solve the problem on the summary. We repeat the latter 50 times and report the average time and risk obtained. For the uniform subsampler, $s_j$ refers to the subsample size, and for the coresets it refers to the size of the coreset. We denote the cumulative running time of summarizing and solving the problem on the summary by $t(n_i, s_j)$ and the obtained risk by $R(n_i, s_j)$. For each procedure, let $\Lambda_{\text{proc}} = \{(n, t(n,s), R(n,s)) \mid n \in \mathscr{N}, s \in \mathscr{S}\}$.

We can now leverage $\Lambda_{\text{proc}}$ to characterize various tradeoffs. For example, to capture the data-time tradeoff for a particular size $n$ we find the minimum running time $t'$ such that $\exists(m, t', R) \in \Lambda_{\text{proc}}$, with $m < n$ and $R \le \varepsilon_{\text{total}}$. Searching $\Lambda_{\text{proc}}$ yields Pareto-optimal boundaries of two oracles: coreset-based (ORACLE-C) and uniform-sampling-based (ORACLE-U). To show that one can navigate the space/time/data/risk tradeoffs in practice using TRAM, we showcase it alongside the oracles in Figure 2. Finding the oracles is computationally prohibitive as it entails a full grid search over $\mathscr{N}$ and $\mathscr{S}$. Nevertheless, the reported times assume the oracles simply *know* what the best summarization is.

**Datasets** SYNTHETIC — We generate synthetic data of 100,000 points in $\mathbb{R}^{100}$ from a mixture of Gaussians. We choose $k = 100$ centers in $[0, 100]^{100}$ and set them as means for the $k$ spherical Gaussian distributions with $\Sigma = 5I$. The relative magnitudes of the clusters are sampled from an exchangeable Dirichlet distribution with parameter $1/20$.

KDD2004BIO — The dataset of the Protein Homology Prediction Task in KDD Cup 2004, with 145,751 instances and 74 attributes that describe the match between two proteins. We fit $k$-means with $k = 150$.

CSN — The Community Seismic Network (CSN) uses smart phones with accelerometers as inexpensive seismometers for earthquake detection. Faulkner et al. [2011] compiled 7 GB of acceleration data and computed 17-dimensional feature vectors. We fit $k$-means with $k = 200$.

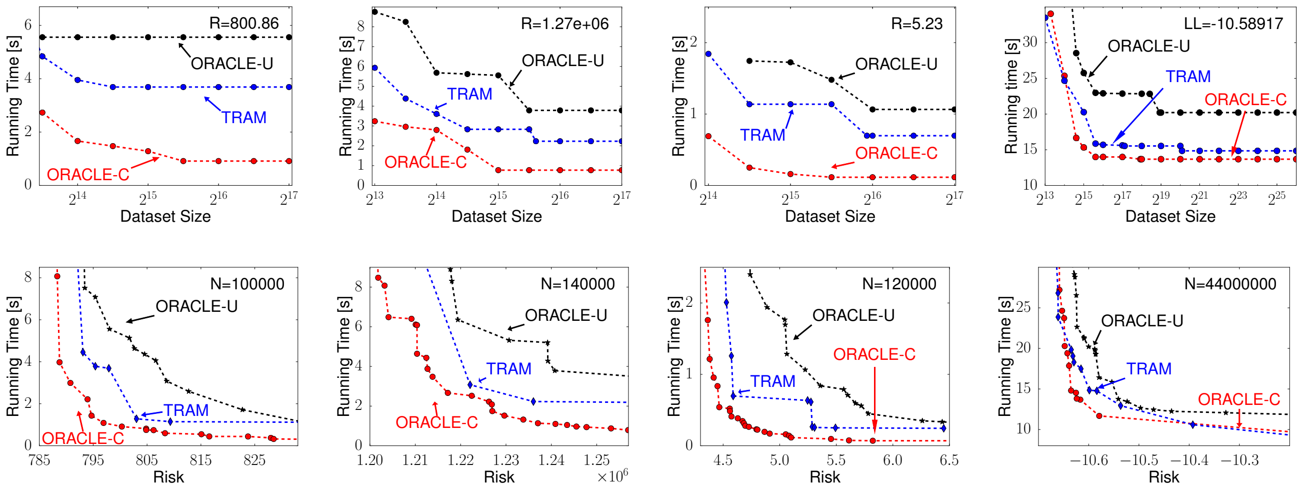


Figure 2: Results for SYNTHETIC, KDD2004BIO, CSN and WEBSCOPE datasets, per column from left to right. Figures in the first row show data-time tradeoffs: best running time for fixed risk tolerance and varying data sizes (cf. Figure 1c). Tradeoffs exist: running time decreases with increasing data size. Furthermore, the coreset procedure dominates uniform subsampling, and TRAM tracks the coreset tradeoff closely, with limited overhead. Figures in the second row show risk-time tradeoffs: best running time for fixed data size and varying risk tolerance (cf. Figure 1d).

YAHOO! WEBSCOPE R6A — 45,811,883 instances in $\mathbb{R}^6$ that represent the user click log displayed on the Yahoo! Front Page. We fit a GMM with $k = 200$ components. The risk here is the negative log likelihood on the hold-out data.

**Parameters** For the $k$-means clustering problem we use the coreset construction from Feldman and Langberg [2011], and a weighted variant of the $k$-means++ algorithm to solve the problem on the subsample. In the case of GMMs, we use the coreset construction from Feldman et al. [2011] and a weighted EM for GMMs. We consider summarization sizes between 100 and 20,000. For TRAM, we start with summarization size and truncation size inversely proportional to the risk required. At every iteration, we double the truncation size and take 1.5-fold ($\beta = -\log_2 1.5$) of the summarization size. $1/5^{\text{th}}$ of the data is assigned to validation, with a $\delta$ of 0.1.

**Observations** The plots in the first row in Figure 2 show the Pareto-optimal boundary for a fixed risk as data size varies. There is a data-time tradeoff as predicted from theory. Furthermore, TRAM traces the solutions achieved by the coreset oracle, implying that we *can* navigate tradeoff curves without oracles. Remarkably, TRAM remains better than the uniform subsampler oracle, eventhough either oracle takes orders of magnitude more time to obtain by exhaustive search. The second row illustrates the existence of a risk-time tradeoffs also: for fixed data size, the time to guarantee a desired risk decreases as the risk increases. Another perspective to these results is as follows. A potential practitioner is faced with three options: solving the problem on the whole dataset or doing so after summarizing, either by truncating to a portion deemed adequate or by strategically summarizing the data with a somewhat larger portion. The former is often out of the question (in the case of GMMs, it may take weeks). Summarization slashes this time down (minutes instead of weeks). However, because the coreset procedure can achieve a faster time even as it accesses a larger portion, it will be more likely to guarantee a desired risk, as compared to the uniform subsampler, at least for *interesting* (small) risk levels.

## 8 CONCLUSIONS

We explored space/time/data/risk tradeoffs achievable via coreset-based data-summarization. Our theory predicts and our empirical results demonstrate the existence and utility of such tradeoffs. We further showed how such tradeoffs can be practically realized via a novel algorithm, TRAM. While our analysis focused on $k$-means, our insights are more generally applicable. In particular, we empirically demonstrated tradeoffs in learning Gaussian Mixture Models. Approaches that optimize cost functions related to the quantization error, such as small-variance limits of non-parametric Bayesian models Jiang et al. [2012], may also immediately benefit from our results. We thus strongly believe that our results present an important step towards understanding tradeoffs in large-scale unsupervised learning. Lastly, given promising summarization-style techniques Pavlov et al. [2000], Bakir et al. [2004], Tsang et al. [2005], similar results may also be possible in supervised learning.

### Acknowledgements

This work was supported in part by ERC StG 307036, an ETH Fellowship, a Microsoft Research Faculty Fellowship, and the Zurich Information Security Center. It was carried out when the second author was an MSR-Inria postdoctoral researcher and an ERCIM "Alain Bensoussan" fellow, funded in part by EU FP7/2007-2013 Grant 246016.

## References

Pankaj K Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30, 2005.

András Antos, László Györfi, and András György. Individual convergence rates in empirical vector quantizer design. *Information Theory, IEEE Transactions on*, 51 (11):4013–4022, 2005.

Gökhan H. Bakir, Léon Bottou, and Jason Weston. Breaking SVM Complexity with Cross-Training. In *NIPS*, 2004.

Maria-Florina Balcan, Steven Ehrlich, and Yingyu Liang. Distributed *k*-means and *k*-median clustering on general topologies. In *NIPS*, pages 1995–2003, 2013.

Shai Ben-David. A framework for statistical clustering with constant time approximation algorithms for k-median and k-means clustering. *Machine Learning*, 66 (2-3):243–257, 2007.

Aharon Birnbaum and Shai S Shwartz. Learning halfspaces with the zero-one loss: time-accuracy tradeoffs. In *NIPS*, pages 935–943, 2012.

Léon Bottou and Olivier Bousquet. The Tradeoffs of Large-Scale Learning. In *NIPS*, volume 20, pages 161–168. NIPS Foundation, 2008.

Venkat Chandrasekaran and Michael I Jordan. Computational and statistical tradeoffs via convex relaxation. *PNAS U.S.A.*, 110(13):E1181–90, March 2013.

Scott E Decatur, Oded Goldreich, and Dana Ron. Computational sample complexity. *SIAM Journal on Computing*, 29(3):854–879, 2000.

Matthew Faulkner, Michael Olson, Rishi Chandy, Jonathan Krause, K Mani Chandy, and Andreas Krause. The next big one: Detecting earthquakes and other rare events from community-based sensors. In *IPSN*, pages 13–24, 2011.

Dan Feldman and Michael Langberg. A Unified Framework for Approximating and Clustering Data. In *STOC*, pages 569–578. ACM, 2011.

Dan Feldman, Morteza Monemizadeh, and Christian Sohler. A PTAS for k-means clustering based on weak coresets. In *Proceedings of the 23rd Annual Symposium on Computational Geometry*, pages 11–18. ACM, 2007.

Dan Feldman, Andreas Krause, and Matthew Faulkner. Scalable training of mixture models via coresets. In *NIPS*, pages 2142–2150, 2011.

Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for *k*-means, PCA and projective clustering. In *SODA*, 2013.

Siegfried Graf and Harald Luschgy. *Foundations of Quantization for Probability Distributions*. Springer, 2000.

Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *STOC*, pages 291–300. ACM, 2004.

Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering. In *ACM SoCG*, pages 332–339. ACM, 1994.

Ke Jiang, Brian Kulis, and Michael I Jordan. Small-variance asymptotics for exponential family Dirichlet process mixture models. In *NIPS*, pages 3167–3175, 2012.

Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):881–892, 2002.

Tamás Linder. Learning-theoretic methods in vector quantization. In *Principles of nonparametric learning*, pages 163–210. Springer, 2002.

Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

Adam Meyerson, Liadan O'Callaghan, and Serge Plotkin. A k-Median algorithm with running time independent of data size. *Machine Learning*, 2004. ISSN 0885-6125.

Dmitry Pavlov, Darya Chudova, and Padhraic Smyth. Towards scalable support vector machines using squashing. In *KDD*, pages 295–299, 2000.

David Pollard. Strong consistency of k-means clustering. *Ann. of Statistics*, 9(1):135–140, 1981.

Rocco A Servedio. Computational sample complexity and attribute-efficient learning. In *STOC*, pages 701–710. ACM, 1999.

Shai Shalev-Shwartz and Nathan Srebro. SVM optimization: inverse dependence on training set size. In *ICML*, pages 928–935, 2008.

Matus Telgarsky and Sanjoy Dasgupta. Moment-based Uniform Deviation Bounds for *k*-means and Friends. *CoRR*, 2013.

Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core vector machines: fast SVM training on very large data sets. *JMLR*, 6:363–392, 2005.