Scalable Deletion-Robust Submodular Maximization: Data Summarization with Privacy and Fairness Constraints

Ehsan Kazemi [1] **Morteza Zadimoghaddam** [2] **Amin Karbasi** [1]

Abstract

Can we efficiently extract useful information from a large user-generated dataset while protecting the privacy of the users and/or ensuring fairness in representation? We cast this problem as an instance of a deletion-robust submodular maximization where part of the data may be deleted or masked due to privacy concerns or fairness criteria. We propose the first memory-efficient centralized, streaming, and distributed methods with constant-factor approximation guarantees against *any* number of adversarial deletions. We extensively evaluate the performance of our algorithms on real-world applications, including (i) Uber-pick up locations with location privacy constraints; (ii) feature selection with fairness constraints for income prediction and crime rate prediction; and (iii) robust to deletion summarization of census data, consisting of 2,458,285 feature vectors. Our experiments show that our solution is robust against even 80% of data deletion.

1. Introduction

It has long been known that solutions obtained from optimization methods can demonstrate striking sensitivity to the parameters of the problem (Bertsimas et al., 2011). Robust optimization, in contrast, is a paradigm in the mathematical programming community with the aim of safeguarding the solutions from the changes in the underlying parameters.

In this paper, we consider submodular maximization, a very well studied discrete optimization problem defined over a finite set of items (e.g., images, videos, blog posts, sensors, etc). Submodularity formalizes the notion of diminishing returns, stating (informally) that selecting an item earlier results in a higher utility than selecting it later. This notion has found far-reaching applications in machine learning (Bach et al., 2013), web search and mining (Borodin et al., 2017), social network (Kempe et al., 2003), crowdsourcing (Singla et al., 2016), and user modeling (Yue & Guestrin, 2011), to name a few. However, almost all the existing methods for submodular maximization, ranging from centralized (Nemhauser et al., 1978; Feldman et al., 2017) to streaming (Badanidiyuru et al., 2014; Feldman et al., 2018), to distributed (Mirzasoleiman et al., 2013; Mirrokni & Zadimoghaddam, 2015; Barbosa et al., 2015), rely on greedy selection of elements. As a result, the returned solution of such methods are remarkably sensitive to even a *single* deletion from the set of items.

The need for efficient deletion-robust optimization methods is wide-spread across many data-driven applications. With access to big and massive data (usually generated by millions of users), along with strong machine learning techniques, many service providers have been able to exploit these new resources in order to improve the accuracy of their data analytics. At the same time, it has been observed that many such inference tasks may leak very sensitive information about the data providers (i.e., personally identifiable information, protected health information, legal or financial data, etc). Similarly these algorithms can encode hidden biases that disproportionately and adversely impact members with certain characteristics (e.g., gender and race).

In order to reduce the effect of information extraction on privacy and fairness, one needs to be able to remove sensitive data points (e.g., geolocations) or discard sensitive data features (e.g., skin color) from the dataset without incurring too much loss in performance. For instance, Article 17 of European "General Data Protection Regulation" states obligations with respect to providing individuals with the "Right to erasure (or Right to be forgotten)". By exercising this right, individuals may enforce the service providers to delete their personal data or put restrictions from using part of it. Similarly, Title VII of the Civil Rights Act of American anti-discrimination law prohibits employment discrimination against certain characteristics (such as color and sex). Thus, to obtain fairer machine learning algorithms, we need to reduce the bias inherent in the training examples due to the lack of certain types of information, not being representative, or reflecting historical biases. This can be done by either removing protected attributes from training data

[1]Department of Computer Science, Yale University, New Haven, Connecticut, USA [2]Google Research, Zurich, Switzerland. Correspondence to: Ehsan Kazemi <ehsan.kazemi@yale.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

(Zemel et al., 2013) or train them separately for different protected groups (Chayes, 2017), among other procedures. Unfortunately, sensitive features or biased data usually are not known a priori and we might be aware of their existence just after training our models (Beutel et al., 2017). Retraining a machine learning model from scratch, after removing sensitive features and biased data, is quite expensive for large datasets. Deletion-robust submodular maximization can save a lot of time and computational resources in these scenarios. In this paper, we provide a computationally feasible way of rerunning the algorithms should some attributes or data points be discarded.

Most existing submodular maximization methods, often used for data extraction (Mirzasoleiman et al., 2013) and informative subset selection (Wei et al., 2015), do not provide such guarantees. In this paper, we develop the first scalable and memory-efficient algorithms for maximizing a submodular function subject to a cardinality constraint that are robust against any number of *adversarial* deletions. This is in sharp contrast to previous methods that could only handle a fixed number of deletions (Orlin et al., 2016; Bogunovic et al., 2017) or otherwise their memory requirement scales multiplicatively with the number of deletions (Mirzasoleiman et al., 2017).

Our contributions: For a monotone submodular function with a cardinality constraint k, we develop the following randomized algorithms that are robust against any d deletions:

1. Centralized: We propose ROBUST-CENTRALIZED that achieves $(1/2 - \delta)$-approximation guarantee (in expectation) with the memory requirement $O\left(k + d \log k/\delta^2\right)$. Note that the memory complexity is only a logarithmic factor (e.g., $\log k$) away from a trivial lower bound $O(k + d)$.

2. Streaming: We propose ROBUST-STREAMING that achieves $(1/2 - \delta)$-approximation guarantee (in expectation) with the memory requirement $O\left(k \log k/\delta + d \log^2 k/\delta^3\right)$.

3. Distributed: We propose ROBUST-DISTRIBUTED that achieves $(0.218 - \delta)$-approximation guarantee (in expectation) with the memory requirement $O\left(m(k + d \log k/\delta^2)\right)$, where m is the number of machines. We also introduce COMPACT-DISTRIBUTED, a variant of ROBUST-DISTRIBUTED, where its memory requirement is independent of number of machines.

Table 1 compares our proposed methods with previous algorithms. **The proofs of all the theoretical results are deferred to the Supplementary Material.**

2. Related Work

Monotone submodular maximization under cardinality constraints is studied extensively in centralized, streaming and distributed scenarios. The classical result of Nemhauser et al. (1978) proves that the simple GREEDY algorithm that starts with an empty set and iteratively adds elements with the highest marginal gain provides $(1 - 1/e)$-approximation guarantee. To scale to large datasets, several streaming algorithms with constant factor approximations have recently been proposed (Badanidiyuru et al., 2014; Kumar et al., 2015; Buchbinder et al., 2015). Also, different distributed submodular maximization algorithms have been developed lately (Mirzasoleiman et al., 2013; Mirrokni & Zadimoghaddam, 2015; Barbosa et al., 2015).

Krause et al. (2008) introduced the robust formulation of the classical cardinality constrained submodular maximization for the first time and gave a bi-criterion approximation to the problem of $\max_{|A|\leq k} \min_{i\in\{1,\cdots,\ell\}} f_i(A)$, where f_i is normalized monotone submodular for every i. Note that submodular maximization of function f that is robust to the deletion of d items can be modeled as a special case of this problem: $\max_{|A|\leq k} \min_{|D|\leq d} f(A \setminus D)$. Krause et al. (2008) guaranteed a robust solution by returning a set whose size is $k(1 + \Theta(\log(dk \log n))$. There are two main drawbacks with this approach when applied to deletions: first, the size of final solution is logarithmically larger than k, and second, the running time is exponential in d. Orlin et al. (2016) designed a centralized algorithm that outputs a set of cardinality k in a polynomial time. Their algorithm is robust to the deletion of only $o(\sqrt{k})$ elements. Bogunovic et al. (2017) further improved the result of Orlin et al. (2016) to $o(k)$ deletions. The approximation guarantees for both of these algorithms are 0.387. The aforementioned methods try to construct a solution without allowing to update the answer after deletion. In contrast, Mirzasoleiman et al. (2017) developed a streaming algorithm which is robust to the deletion of any number of d elements. They keep a set of size $O(kd \log k/\delta)$, and after each deletion they find a feasible solution of size at most k from this set. They also improved the approximation guarantee to $1/2 - \delta$. The main drawback of this algorithm is the memory requirement, which is quite impractical for large values of d and k; e.g., for $k = O(\sqrt{n})$ and $d = O(\sqrt{n})$ the memory requirement is even larger than n. Independently and concurrently with our work, Mitrovic et al. (2017) presented a robust to deletion streaming algorithm. Also, there are several recent works on robust optimization of non-submodular functions (Bogunovic et al., 2018; Tzoumas et al., 2018).

Submodular maximization has been widely used in classical machine learning and data mining applications, including extracting representative elements with exemplar based clustering (Krause & Gomes, 2010), data summarization through active set selection (Herbrich et al., 2003; Seeger, 2004), feature selection (Krause & Guestrin, 2005) and document summarization (Lin & Bilmes, 2011).

3. Problem Definition

Assume we have a set function $f : 2^V \to \mathbb{R}_{\geq 0}$. We define the marginal gain of an element $e \in V$ to the set $A \subseteq V$

Table 1. Comparison of algorithms for robust submodular maximization with a cardinality constraint k. Randomized algorithms for which the bounds hold in expectation are marked (R).

Algorithm	Max. Robustness	Approx.	Memory	Setup
OSU (Orlin et al., 2016)	$o(\sqrt{k})$	0.387	k	Centralized
PRO-GREEDY (Bogunovic et al., 2017)	$o(k)$	0.387	k	Centralized
ROBUST (Mirzasoleiman et al., 2017)	arbitrary d	1/2 - δ	$O(kd \log k/\delta)$	Streaming
STAR-T-GREEDY (Mitrovic et al., 2017)	arbitrary d	$\frac{0.149}{1+\delta}(1 - {}^{1}/{\lceil \log k \rceil})$	$O(k \log^2 k/\delta + d \log^3 k/\delta)$	Streaming
ROBUST-CENTRALIZED (R) (ours)	arbitrary d	$1/2 - \delta$	$O(k + d \log k/\delta^2)$	Centralized
ROBUST-STREAMING (R) (ours)	arbitrary d	$1/2 - \delta$	$O(k \log k/\delta + d \log^2 k/\delta^3)$	Streaming
ROBUST-DISTRIBUTED (R) (ours)	arbitrary d	$0.218 - \delta$	$O(m(k + d \log k/\delta^2))$	Distributed
COMPACT-DISTRIBUTED (R) (ours)	arbitrary d	$0.109 - \delta$	$O(k + d \log k/\delta^2)$	Distributed

by $\Delta_f(e|A) = f(A \cup \{e\}) - f(A)$. The function f is submodular if for all $A \subseteq B \subseteq V$ and $e \in V \setminus B$, we have $\Delta_f(e|A) \geq \Delta_f(e|B)$. A submodular function f is monotone if for every $A \subseteq B \subseteq V$, we have $f(A) \leq f(B)$.

In many submodular optimization applications, a subset of items of the ground set V may be removed at different points in time. For this reason, we require to find solutions which are robust to the deletion. Indeed, the goal is to maximize a submodular function f over a set V of items under a cardinality constraint k, where it is robust to the deletion of any subset $D \subset V$ of size $|D| \leq d$. More precisely, we are interested in solving the following problem for each possible (and unknown a priori) instance of D:

$$S^* = \underset{S \subseteq V \setminus D, |S| \leq k}{\arg\max} \; f(S). \tag{1}$$

We also define OPT $= f(S^*)$. The most straightforward approach to this problem is to solve Eq. (1) for each instance of D. Unfortunately, solving Eq. (1), for large datasets, is computationally prohibitive. Also, deletion of elements from the set V can happen at different stages in real time applications. This makes the problem even harder. Our solution to this problem is to maintain a small set $A \subset V$, called a core-set of V, where for each set D we can efficiently find a subset $B \subseteq A \setminus D$ that provides an acceptable approximation for Eq. (1). Note that set A is constructed without knowing set D. For this reason, next we define the notion of (α, d)-robust randomized core-set.

Definition 1. *A random subset of $A \subseteq V$ is an (α, d)-robust randomized core-set for a set V, if for any subset $D \subseteq V$ of size $|D| \leq d$, there exists a $B \subseteq A \setminus D, |B| \leq k$ such that*

$$\mathbb{E}[f(B)] \geq \alpha \cdot \max_{S \subseteq V \setminus D, |S| \leq k} f(S),$$

where expectation is taken over the randomization of set A.

4. Robustness and Cardinality Constraint

In this section, we present three fast and scalable randomized algorithms. These algorithms solve the problem of robust submodular maximization in centralized, streaming and distributed scenarios. Our algorithms provide, in expectation, constant factor approximation guarantees, where they are robust to the (even adversarial) deletion of any d items from the set V. In our setting, an adversary might try to find a set of inputs for which our algorithms fail to provide good results. In order to make the optimization robust to the adversarial deletions, we introduce randomness in the selection process. We assume that the adversary does not have access to the random bits of the randomized algorithms.

The proposed algorithms are designed based on a general idea that the elements are chosen randomly from a large enough pool of *similar* items. This idea is useful because the adversary is not aware of the random bits of the algorithms, which makes the deletion probability of elements we have chosen negligible. Therefore, we can bound the expected value of each selected set.

Our solution consists of two steps. In the first step, we find a small core-set of elements (in comparison to the whole dataset). We prove that after the deletion of at most d arbitrary elements, we can still find a good approximation for the optimization problem in this small set. In the second step, we choose at most k elements from the core-set we have found in the first step. We prove a constant approximation factor for our algorithm in expectation. This guarantees that the core-set is (α, d)-robust randomized for a constant α and arbitrary d.

In the optimization procedure, we use a thresholding idea to select elements. Similar ideas have been used previously for designing streaming algorithms (Badanidiyuru et al., 2014; Buchbinder et al., 2015; Chekuri et al., 2015). In those algorithms, when an element of the stream arrives, if this element has *sufficiently* large marginal value it is kept; otherwise it is discarded. In the robust submodular maximization, we keep a large enough pool of elements with sufficient marginal values before adding or discarding them. We randomly pick an element when the size of pool is at least d/ϵ. Thus the element picked at each step is deleted with a probability at most ϵ. This is true because the size of deleted items is at most d. To guarantee the quality of the chosen elements after the deletion (i.e., we want the expected value of f over the set of picked elements does not change a lot after deletion), not only they should have been picked from a large pool of elements, the elements of pool should have almost the same marginal gains. To explain, in more details, why we need this property consider the example in Appendix A.

4.1. Centralized Algorithm

In this section we outline a centralized algorithm, called ROBUST-CORESET-CENTRALIZED, to find an (α, d)-robust core-set. We also present the ROBUST-CENTRALIZED algorithm which is able to find a good solution from the core-set.

Badanidiyuru et al. (2014) showed that one way to obtain a constant factor approximation to the classical submodular maximization problem is to use a thresholding idea. They proved that choosing elements with marginal gain at least $\tau^* = \frac{\text{OPT}}{2k}$ from a stream until a maximum of k elements are chosen returns a set with an approximation factor of $1/2$. The main problem with this primary idea is that the value of OPT is not known by the algorithm. Badanidiyuru et al. (2014) pointed out that, from the submodularity of f, we have $\Delta_0 \leq \text{OPT} \leq k\Delta_0$ where Δ_0 is the largest value in set $\{f(\{e\})|e \in V\}$. By dividing the range $[\Delta_0, k\Delta_0]$ into intervals of $[\tau_i, \tau_{i+1})$ (where τ_{i+1}/τ_i is close to 1) it is possible to find a good enough approximation for OPT.

We should first note that due to the deletion process, the relevant maximum singleton value is not Δ_0 anymore, and it is $\Delta'_0 = \max_{e \in V \setminus D} f(\{e\})$. The algorithm is unaware of set D, therefore Δ'_0 could be anywhere in the range $[\Delta_d, \Delta_0]$ where Δ_d is the $(d+1)$-th largest value in the set $\{f(\{e\})|e \in V\}$. The lower bound of Δ_d is implied by the fact that at most d elements will be deleted. So $\tau^* = \frac{\text{OPT}}{2k}$ could fall anywhere in the range $[\Delta_d/2k, \Delta_0]$. Unlike the deletion free case, the upper and lower limits of this range do not differ only by a multiplicative factor of k, thus a naive approach makes us try arbitrarily large number of different choices to find a good estimate of τ^*. We resolve this issue by the following observation.

We reserve a set B of elements that might be valuable after the deletion process. Let V_d be the $(d+1)$ largest singleton value elements, i.e., the top $d+1$ elements in the set $\{f(\{e\})|e \in V\}$. We preserve all elements of V_d for the next round by inserting them to B. This way, we do not have to worry about thresholds above Δ_d as all elements that might have marginal value above Δ_d to any set should be in set V_d and they are added to B. Therefore, we consider all thresholds in the set $\mathrm{T} = \{(1+\epsilon)^i | \frac{\Delta_d}{2k} \leq (1+\epsilon)^i \leq \Delta_d\}$. Starting from the largest $\tau \in \mathrm{T}$ to the smallest, we iteratively construct two sets A_τ and B_τ. At the end of the algorithm, the set B is defined as the union of V_d and $\cup_{\tau \in \mathrm{T}} B_\tau$. We output set B, along with all sets $\{A_\tau\}_{\tau \in \mathrm{T}}$, as the core-set.

We initialize A_τ to $\varnothing$. We let B_τ to be the set of elements whose marginal values to the set $\cup_{\tau' \geq \tau} A_{\tau'}$ is in the range $[\tau, (1+\epsilon)\tau)$. We note that this is a dynamic definition and whenever we add an element to any of A_τ sets, the related B_τ set might change as well. Elements in the set B_τ are similar to each other in terms of their marginal values. Without deletions, we can choose any element from B_τ and add it to our solution. However, if B_τ has only a few elements, the adversary can delete all of them, and we will be left with an arbitrary poor solution. To make the selection process robust, we select a random element from B_τ and add it to A_τ only if there are at least d/ϵ elements in B_τ. This way even if all the deleted elements are from the set B_τ, the probability of each selected element being deleted is at most ϵ. We also know that all elements added to A_τ have similar marginal values and are interchangeable. We keep adding elements to A_τ until either $\cup_{\tau' \geq \tau} A_{\tau'}$ has k elements or the size of set B_τ becomes smaller than d/ϵ. At this stage, we keep both sets A_τ and B_τ as a part of the output core-set. We also remove them from the ground set V and move on to the next lower threshold. The pseudo code of ROBUST-CORESET-CENTRALIZED is given in Algorithm 1.

Algorithm 1 ROBUST-CORESET-CENTRALIZED

```
 1: $\Delta_d \leftarrow$ the $(d+1)$-th largest value of $\{f(\{e\})|e \in V\}$
 2: $V_d \leftarrow$ all the $d+1$ elements with the largest values in
    set $\{f(\{e\})|e \in V\}$
 3: $\mathrm{T} = \{(1+\epsilon)^i | \frac{\Delta_d}{2(1+\epsilon)k} \leq (1+\epsilon)^i \leq \Delta_d\}$
 4: For each $\tau \in \mathrm{T}$ : $\{A_\tau\} \leftarrow \varnothing$ and $\{B_\tau\} \leftarrow \varnothing$
 5: $V \leftarrow V \setminus V_d$
 6: for $\tau \in \mathrm{T}$ from the highest to the lowest do
 7:   while $|B_\tau| \geq d/\epsilon$ for $B_\tau = \{e \in V : \tau \leq$
      $\Delta_f(e|\cup_{\tau' \geq \tau} A_{\tau'}) < (1+\epsilon)\tau\}$ and $|\cup_{\tau' \geq \tau} A_{\tau'}| < k$
      do
 8:     Randomly pick an element $e$ from $B_\tau$ and add it
        to $A_\tau$, i.e., $A_\tau \leftarrow A_\tau \cup \{e\}$
 9:   $V \leftarrow V \setminus (A_\tau \cup B_\tau)$
10: $B \leftarrow \{\cup B_\tau\} \cup V_d$
11: Return $\{A_\tau\}, B$
```

The sets $\{A_\tau\}$ and B are the outputs (core-set) of ROBUST-CORESET-CENTRALIZED. In Appendix B , we show how ROBUST-CENTRALIZED (with pseudo code given in Algorithm 2) returns a solution for submodular maximization problem after the deletion of set D.

Theorem 1. *For any $\delta > 0$, by setting $\epsilon = \frac{2\delta}{3}$,* ROBUST-CORESET-CENTRALIZED *and* ROBUST-CENTRALIZED *satisfy the following properties:*

- ROBUST-CENTRALIZED *outputs a set S such that $|S| \leq k$ and $\mathbb{E}[f(S)] \geq (1/2 - \delta) \cdot \text{OPT}$.*
- ROBUST-CORESET-CENTRALIZED *outputs at most $O\left(k + d\log k/\delta^2\right)$ elements as the core-set.*
- *The query complexities of* ROBUST-CORESET-CENTRALIZED *and* ROBUST-CENTRALIZED *are $O\left((k + \log k/\delta)|V|\right)$ and $O\left((k + d\log k/\delta^2)(\log k/\delta)\right)$.*

Algorithm 2 ROBUST-CENTRALIZED

```
1: Input: $\{A'_\tau\}$ and $B'$ {$A'_\tau$ and $B'$ contain elements of $A_\tau$ and $B$ (outputs of ROBUST-CORESET-CENTRALIZED) after deletion.}
2: Output: Set $S$ of cardinality at most $k$
3: $\Delta'_0 \leftarrow$ the largest value of $\{f(\{e\})|e \in \{\cup A'_\tau\} \cup B'\}$
4: $\mathrm{T}' = \{(1+\epsilon)^i | \frac{\Delta'_0}{2(1+\epsilon)k} \leq (1+\epsilon)^i \leq \Delta'_0\}$
5: for $\tau \in \mathrm{T}'$ from the highest to the lowest do
6:     $S_\tau \leftarrow \bigcup_{\tau' \in \mathrm{T}', \tau' \geq \tau} A'_{\tau'}$
7:     for all $e \in B'$ do
8:         if $\Delta_f(e|S_\tau) \geq \tau$ and $|S_\tau| < k$ then
9:             $S_\tau \leftarrow S_\tau \cup e$
10: Return $\arg\max_{S_\tau} f(S_\tau)$
```

4.2. Streaming Algorithm

In many applications, the dataset does not fit in the main memory of a single machine or even the data itself arrives as a stream. So it is not possible to use centralized algorithms which need random access to the whole data. In this section, we present a streaming algorithm with a limited available memory. We first use the thresholding idea of Section 4.1 in order to find a core-set for V. Then we show that it is possible to find a good solution from this core-set when deletion happens. Recall that for ROBUST-CORESET-CENTRALIZED, the maximum singleton element and the thresholds are fixed while in the streaming setting, they may change as new elements arrive. To apply ideas of the centralized algorithm, we should overcome the following challenges: (i) it is not possible to make several passes over the data for different thresholds (i.e., we cannot start from the largest possible marginal gain to the lowest), and (ii) the value of Δ_0 and Δ_d are not known a priori.

We show that it is possible to maintain a good approximation of OPT even with a single pass over the data. From now on, let Δ_0 and Δ_d, respectively, denote the largest and the $(d+1)$-th largest singleton values in the stream of data at time step t. First, note that $\Delta_d \leq$ OPT and the marginal gain of all the currently received elements is at most Δ_0. Therefore, it is enough to consider thresholds in the range $[\frac{\Delta_d}{2k}, \Delta_0]$. A new threshold is instantiated when the maximum singleton element is changed. These new (increasing) thresholds are between the current maximum and the previous one. Therefore, all the elements with marginal gains larger than the new threshold will appear after its instantiation.

ROBUST-CORESET-STREAMING, for each threshold τ, keeps two sets A_τ and $B_\tau = \cup_{\tau' \geq \tau} B_{\tau,\tau'}$. All the elements with marginal gains at least τ to set A_τ are *good enough* to be picked by this instance of the algorithm. In order to make the selected elements robust to deletions, we should put all good enough elements in different $B_{\tau,\tau'}$ sets, with thresholds τ' in the range $[\tau, \Delta_0]$, based on their marginal values. Whenever a set $B_{\tau,\tau'}$ becomes large, we pick one

Algorithm 3 ROBUST-CORESET-STREAMING

```
1: $\mathrm{T} = \{(1+e)^i | i \in \mathbb{Z}\}$
2: For each $\tau, \tau' \in \mathrm{T}$ : $\{A_\tau\} \leftarrow \varnothing$ and $\{B_{\tau,\tau'}\} \leftarrow \varnothing$
3: for every arriving element $e_t$ do
4:     $\Delta_d \leftarrow$ the $(d+1)$-th largest element of $\{f\{e_1\}, \cdots, \{f\{e_t\}\}$
5:     $\Delta_0 \leftarrow$ the largest element of $\{f\{e_1\}, \cdots, \{f\{e_t\}\}$
6:     $\mathrm{T}_t = \{(1+\epsilon)^i | \frac{\Delta_d}{2(1+\epsilon)k} \leq (1+\epsilon)^i \leq \Delta_d\}$
7:     Delete all $A_\tau$ and $B_{\tau,\tau'}$ such the $\tau$ or $\tau' \notin \mathrm{T}_t$
8:     for $\tau \in \mathrm{T}_t$ do
9:         if $|A_\tau| < k$ and $\tau \leq \Delta_f(e|A_\tau)$ then
10:            Add $e_t$ to $B_{\tau,\tau'}$ such that for $\tau' \leq \Delta_f(e_t|A_\tau) < \tau'(1+\epsilon)$
11:            while $\exists \tau''$ such that $|B_{\tau,\tau''}| \geq d/\epsilon$ do
12:                Randomly pick an element $e$ from $B_{\tau,\tau''}$ and add it to $A_\tau$, i.e., $A_\tau \leftarrow A_\tau \cup \{e\}$
13:                For all $e \in \bigcup_{\tau'' \in \mathrm{T}_i, \tau'' \geq \tau} B_{\tau,\tau''}$ recompute $\Delta_f(e|A_\tau)$ and re-place them in correct bins
14: for $\tau \in \mathrm{T}_n$ do
15:     $B_\tau \leftarrow \bigcup_{\tau' \in \mathrm{T}_n, \tau' \geq \tau} B_{\tau,\tau'}$
16: Return $\{A_\tau\}, \{B_\tau\}$
```

element of it randomly to add to A_τ. This ensures that an element is picked from a large pool of almost similar elements. Formally, all the elements with a marginal gain in the range $[\tau', \tau'(1+\epsilon))$ are added to the set $B_{\tau,\tau'}$. When the size of a $B_{\tau,\tau'}$ is at least d/ϵ, we randomly pick an element from $B_{\tau,\tau'}$ and add it to A_τ. Adding an element to A_τ may decrease the marginal gains of elements in $B_{\tau,\tau'}$ sets. So we recompute their marginal gains and put them in the right $B_{\tau,\tau''}$ set (they are kept if their marginal gains are at least τ, otherwise they are discarded). These changes may make another set large, so we keep adding elements to A_τ while we find a large $B_{\tau,\tau''}$ set. This process continues until a maximum of k elements are added to A_τ or the stream of data ends. Note that there are at most d elements with marginal gains in the range $(\Delta_d, \Delta_0]$; we can simply keep these elements (refer to it as set V_d). For all $\Delta_d < \tau \leq \Delta_0$, we have $A_\tau = \varnothing$, because there is no pool of size at least d/ϵ elements to pick from it. Also, for $B_{\tau,\tau'}$ sets, we do not need to cover the range $(\Delta_d, \Delta_0]$ with too many thresholds. Indeed, when Δ_d changes (it can only increase), we can update the set V_d and locate the removed elements from V_d into a correct $B_{\tau,\tau'}$. Therefore, it is sufficient to consider only thresholds in the range$[\frac{\Delta_d}{2k}, \Delta_d]$. The pseudo code of ROBUST-CORESET-STREAMING is given in Algorithm 3.

In Appendix D, we introduce another algorithm (called ROBUST-STREAMING) such that after deletion of any set D from the core-set finds a solution with an expected approximation guarantee of $\frac{1-3\epsilon}{2}$ to the optimum solution.

Theorem 2. *For any $\delta > 0$, by setting $\epsilon = \frac{2\delta}{3}$,* ROBUST-CORESET-STREAMING *and* ROBUST-STREAMING *satisfy the following properties:*

Algorithm 4 ROBUST-DISTRIBUTED

```
1: for $e \in V$ do
2:     Assign $e$ to a machine $i$ chosen uniformly at random;
3: Let $V_i$ be the elements assigned to machine $i$
4: Run ROBUST-CORESET-CENTRALIZED (Algorithm 1) on each machine to obtain $\{A^i_\tau\}$ and $B^i$
5: Run ROBUST-CENTRALIZED (Algorithm 2) on each $\{A^{i\prime}_\tau\}$ and $B^{i\prime}$ to get the set $S^i$ of cardinality at most $k$ from each machine $\{\{A^{i\prime}_\tau\}$ and $B^{i\prime}$ are elements of $\{A^i_\tau\}$ and $B^i$ after deletion of set $D$.$\}$
6: $S \leftarrow \arg\max_{S^i}\{f(S^i)\}$
7: $T \leftarrow$ GREEDY$(\{\bigcup_i \bigcup_{\tau \in \mathrm{T}^i} A^{i\prime}_\tau\} \bigcup \{\bigcup_i B^{i\prime}\})$
8: Return $\arg\max\{f(T), f(S)\}$
```

- ROBUST-STREAMING *outputs a set* S *such that* $|S| \leq k$ *and* $\mathbb{E}[f(S)] \geq (1/2 - \delta) \cdot \mathrm{OPT}$.
- ROBUST-CORESET-STREAMING *makes one pass over the dataset.*
- ROBUST-CORESET-STREAMING *outputs at most* $O\left(k \log k/\delta + d \log^2 k/\delta^3\right)$ *elements as the core-set.*
- *The query complexities of* ROBUST-CORESET-STREAMING *and* ROBUST-STREAMING *are* $O\left(|V| \log k/\delta + dk \log^2 k/\delta^3\right)$ *and* $O\left(d \log^3 k/\delta^4\right)$.

4.3. Distributed Algorithm

In this section, build upon ideas from (Mirzasoleiman et al., 2013; Mirrokni & Zadimoghaddam, 2015; Barbosa et al., 2015), we present a robust distributed submodular maximization algorithm, called ROBUST-DISTRIBUTED. We prove that our distributed algorithm finds an (α, d)-robust randomized core-set with a constant α and any arbitrary d.

ROBUST-DISTRIBUTED is a two-round distributed algorithm within a MapReduce framework. It first randomly partitions dataset between m machines. Each machine i runs ROBUST-CORESET-CENTRALIZED on its data and passes the result (i.e., sets $\{A^i_\tau\}$ and B^i) to a central machine. After the deletion of the set D, this single central machine runs m instances of ROBUST-CENTRALIZED on the outputs received from each machine i and finds solutions S^i. In addition, it runs the classical GREEDY on the union of sets received from all machines (i.e., union of all sets $\{A^{i\prime}_\tau\}$ and $B^{i\prime}$) to find another solution T. The final solution is the best answer among T and sets S^i. ROBUST-DISTRIBUTED is outlined in Algorithm 4.

Theorem 3. *For any* $\delta > 0$, *by setting* $\epsilon = \delta/2$, ROBUST-DISTRIBUTED *outputs a set* S, $|S| \leq k$ *such that* $\mathbb{E}[f(S)] \geq \alpha\beta/(\alpha+\beta) \cdot \mathrm{OPT}$, *where* $\alpha = 1/3 - \delta$ *and* $\beta = 1 - 1/e$. *This results in an approximation factor of* $0.218 - \delta$.

Corollary 1. *Running* ROBUST-CORESET-CENTRALIZED *on the output of* ROBUST-DISTRIBUTED *produces a compact core-set of size* $O\left(k + d \log k/\delta^2\right)$. *Also,* ROBUST-CENTRALIZED *finds a solution with* $(0.109 - \delta)$-*approximation guarantee from this compact core-set. We refer to this version of our distributed algorithm as* COMPACT-DISTRIBUTED.

The main motivation of COMPACT-DISTRIBUTED is that the memory complexity does not increase with the number of machines m (while it still provides a constant factor approximation).

5. Experimental Results

In this section, we extensively evaluate the performance of our algorithms on several publicly available real-world datasets. We consider algorithms that can be robust to the deletion of any number of items and return k elements after deletion. Note that both OSU (Orlin et al., 2016) and PRO-GREEDY (Bogunovic et al., 2017) are robust to the deletion of only $o(k)$ items. For this reason, we compare our proposed methods with three other baselines: (i) ROBUST (Mirzasoleiman et al., 2017), (ii) STAR-T-GREEDY (Mitrovic et al., 2017), and (iii) the stochastic greedy algorithm (Mirzasoleiman et al., 2015) (SG), where we first obtain a solution S of size $r = 6k$ (we set $r > k$ to make the solution robust to deletion), and then we report GREEDY$(S \setminus D)$ as the final answer.

In our experiments, we evaluate the effect of three parameters: (i) d where an algorithm is designed to be robust to d deletions; (ii) cardinality constraint k of the final solution; and (iii) number of deleted elements r. The objective value of all algorithms are normalized to the utility obtained from a classical greedy algorithm that knows the set of deleted items D beforehand. Note that we are able to guarantee the performance of our algorithms (also this is true for ROBUST (Mirzasoleiman et al., 2017) and STAR-T-GREEDY (Mitrovic et al., 2017)) only when the number of deletions r is less than d. While the theoretical improvements of our algorithms for larger values of d is more significant (see Table 1), for a fair comparison, we used the experimental setting of Mirzasoleiman et al. (2017). In these experiments, we also evaluate the effect of larger number of deletions, i.e., where $r \gg d$. We observe, even though our algorithms are not designed for such higher number of deletions, they demonstrate a gracefully robust behavior.

5.1. Location Privacy

In a wide range of applications, data can be represented as a kernel matrix K, which encodes the similarity between different items in the database. In order to find a representative set S of cardinality k, a common objective function is

$$f(S) = \log\det(I + \alpha K_{S,S}), \tag{2}$$

where $K_{S,S}$ is the principal sub-matrix of K indexed by S and $\alpha > 0$ is a regularization parameter (Herbrich et al., 2003; Seeger, 2004; Krause & Guestrin, 2005). This function is monotone submodular.

In this section, we analyze a dataset of 10,000 geolocations.

Each data entry is longitude and latitude coordinates of Uber pickups in Manhattan, New York in April 2014 (Uber-Dataset). Our goal is to find k representative samples using the objective function described in Eq. (2). The similarity of two location samples i and j is defined by a Gaussian kernel $K_{i,j} = \exp(-d_{i,j}^2/h^2)$, where the distance $d_{i,j}$ (in meters) is calculated from the coordinates and h is set to 5000. We set $d = 5$, i.e., we make algorithms (theoretically) robust to deletion of at most five elements. To compare the effect of deletions on the performance of algorithms, we use two strategies to choose the deleted items: (i) classical greedy algorithm, and (ii) the stochastic greedy algorithm.

In the first experiment, we study the effect of deleting different number of items on the normalized objective values. To refer to an algorithm with a specific deletion strategy, we use the name of algorithm followed by the deletion strategy, e.g., Rob-Stream-G refers to ROBUST-STREAMING where the deleted items are picked by greedy strategy. From Fig. 1a, we observe that ROBUST-STREAMING and ROBUST-CENTRALIZED are more robust to deletion than ROBUST and SG. The effect of deleting by greedy strategy on the performance of algorithms is more pronounced than SG strategy. It can be seen that, even by deleting more than $d = 5$ items, our algorithms maintain their performance. Also, SG (which is not designed to be robust to deletions) shows the worst performance.

Other than normalized objective values, the memory requirement of each algorithm is quite important. Indeed, we are interested in deletion-robust algorithms that do not keep many items. Fig. 1b compares the memory complexity of algorithms. We observe that ROBUST-CENTRALIZED needs to keep the least number of items. For ROBUST algorithm, the memory complexity increases super linear in k (it is $O(k \log k)$), which makes it quite impractical for large values of k and d. Also, we observe ROBUST-STREAMING outperforms STAR-T-GREEDY in both objective function and memory requirement. To sum-up, we observe that our algorithms provide the best of two worlds: while their normalized objective values are clearly better than other baselines, they need to keep much fewer number of items.

5.2. Submodular Feature Selection

One of the challenges in learning from high dimensional data is to select a subset of relevant features in a computationally feasible way. For this reason, the quality of a subset of features S can be captured by the mutual information between attributes in S and the class variable Y (Krause & Guestrin, 2005). More specifically,

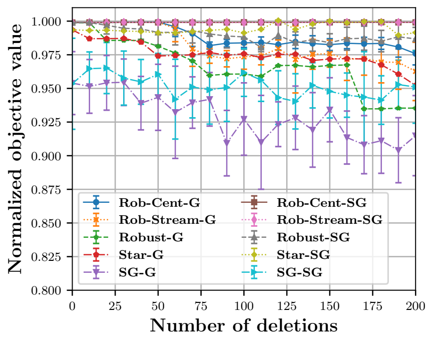
$$I(Y; X_S) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}_S} p(x, y) \log_2 \left(\frac{p(x, y)}{p(x)p(y)} \right),$$

where X_S is a random variable that represents the set S of k features. The joint distribution on $(Y, X_1, \cdots, X_k)$, under the Naive Bayes assumption, is defined by $p(y, x_1, \cdots, x_k) = p(y) \prod_{i=1}^{k} p(x_i|y)$. This assumption makes the computation of joint distribution tractable. In our experiments, we estimate each $p(x_i|y)$ by counting frequencies in the dataset. In the feature selection problem, the goal is to choose k features such that maximizing $f(S) = I(Y; X_S)$. It is known that the function $f(S) = I(Y; X_S)$, under the Naive Bayes assumption, is monotone submodular (Krause & Guestrin, 2005).

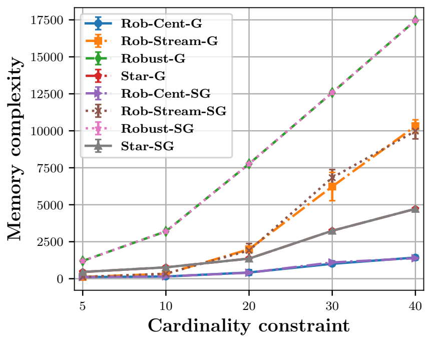
In this section and Appendix G, we use this feature selection method on two real datasets. We first show that our algorithms, after the deletion of sensitive features (i.e., features that might cause unfairness in the final classifier) provide results with near optimal quality (based on mutual information). Second, we demonstrate that classifiers that are trained on these selected features perform very well.

In the first experiment, we use the *Adult Income* dataset from UCI Repository (Blake & Merz, 1998). This dataset contains information about 32,561 individuals and whether income of those individuals is over 50K a year. We extract 113 binary features from this dataset. The goal of the classification task is to predict the income status of 16,281 test cases. For the deletions, we remove sensitive features that might result in the unfairness, e.g., features about sex, race, nationality, marital status and relationship status. Fig. 1c compares algorithms based on different number of deletions for $k = 5$ and $k = 10$. We observe that for both values of k, ROBUST-CENTRALIZED considerably outperforms ROBUST (Mirzasoleiman et al., 2017) and SG. Also, the performance of ROBUST is better than SG.

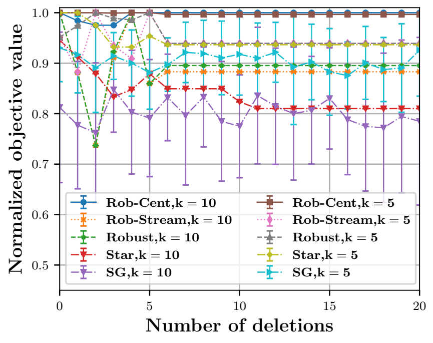
To further investigate the effect of deletions, we compare accuracy of different classifiers, where each is trained on the features found by our algorithms and baselines. We train two type of classifiers: (i) Naive Bayes (Zhang, 2004) and (ii) SVM (Smola & Schölkopf, 2004). From Table 2, we observe that a SVM classifier, which is trained over all features, results in an accuracy of 83.0%. If we use a greedy algorithm to find the best 5 features and train SVM classifier on those features, the accuracy will drop to 79.6% (clearly there is a trade off between the number of features and accuracy). After deleting 10 features that might result in unfairness in classification (e.g., race and sex), we again use the greedy algorithm to find the best five features (referred to as GREEDY$_D$). The accuracy in this case is 79.3%. Interestingly, we observe that the accuracies of classifiers which are trained on the features found by ROBUST-CENTRALIZED and ROBUST-STREAMING drop only by 0.2%. Also, for Naive Bayes classifier, we do not observe any decrease on the accuracy when we train on the features found by our algorithms. Finally, both Centralized (22) and Streaming (29) algorithms need to keep fewer number of items than ROBUST (39) and STAR-T-GREEDY (50).



(a) Uber dataset: we set $d = 5$ and $k = 20$.



(b) Uber dataset: we set $d = 5$ and $r = 100$.



(c) Adult Income: we set $d = 3$

Figure 1. (a) The effect of deletion on the performance of algorithms with respect two different deletion strategies; (b) memory complexity of robust algorithms for different cardinality constraints; (c) The effect of deletion on the performance for feature selection.

Table 2. The comparison of Naive Bayes and SVM classifiers for Adult Income dataset. Ten sensitive features are deleted. The number of stored features is reported in parenthesis.

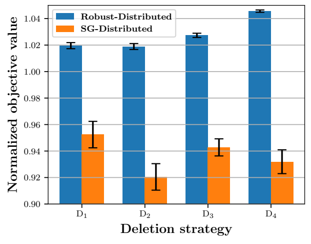
Algorithm	Naive Bayes (Acc.)	SVM (Acc.)
All features	0.798	0.830
GREEDY	0.788	0.796
GREEDY$_D$	0.781	0.793
Rob-Cent	0.781 (22)	0.791
Rob-Stream	0.781 (29)	0.791
ROBUST	0.779 (39)	0.788
STAR-T-GREEDY	0.779 (50)	0.787

5.3. Large Data Summarization

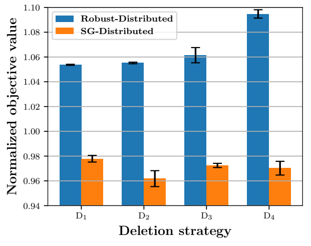
To evaluate the performance of ROBUST-DISTRIBUTED on large datasets, we consider the *Census1990* dataset from UCI Repository (Blake & Merz, 1998). This dataset consists of 2,458,285 data points with 68 features. We are going to find k representative samples from this large dataset. We apply the set selection objective function described in Eq. (2). The similarity between two entries x and x' is defined by $1 - \frac{\|x-x'\|}{\sqrt{68}}$, where $\|x - x'\|$ is the Euclidean distance between feature vectors of x and x'.

We randomly split the dataset into $m = 12$ partitions. For each instance of ROBUST-CORESET-CENTRALIZED, we set $d = 25$ with an $\epsilon = 0.1$. As a baseline, we consider a distributed version of stochastic greedy algorithm (refer to it as SG-DISTRIBUTED). For this algorithm, we first run stochastic greedy on each partitions to select $S_i = 6k$ items. After deletion of D, we report $f(\text{GREEDY}(\cup S_i \setminus D))$ as the final result. Also, we normalize the utility of functions to the objective value of an instance of SG-DISTRIBUTED that knows the set of deleted items D in advance. For deletions, we propose four different strategies: D_1 randomly deletes 50% of items, D_2 randomly deletes 80% of items, D_3 deletes all men in the dataset, and D_4 deletes all women.

We investigate the effect of different deletion strategies for



(a) $k = 50$



(b) $k = 100$

Figure 2. Census1990 dataset: ROBUST-DISTRIBUTED versus SG-DISTRIBUTED for four different deleting strategies.

two values of $k \in \{50, 100\}$. In Figs. 2a and 2b, we observe that ROBUST-DISTRIBUTED clearly outperforms SG-DISTRIBUTED in all cases. Furthermore, we observe that the objective value of ROBUST-DISTRIBUTED in all scenarios is even better than our reference function for normalization (normalized objective values are larger than 1). Each machine on average stores 209.3 (for $k = 50$) and 348.3 (for $k = 100$) items. The standard deviations of memory complexities are 36.9 and 26.5, respectively. To conclude, ROBUST-DISTRIBUTED enables us to robustly summarize a dataset of size 2,458,285 with storing only ≈4500 items. Our experimental results confirm that this core-set is robust to the deletion of even 80% of items.

6. Conclusion

In this paper, we considered the problem of deletion-robust submodular maximization. We provided the first scalable and memory-efficient solutions in different optimization settings, namely, centralized, streaming, and distributed models of computation. We rigorously proved that our methods enjoy constant factor approximations with respect to the optimum algorithm that is also aware of the deleted set of elements. We showcased the effectiveness of our algorithms on real-word problems where part of data should be deleted due to privacy and fairness constraints.

Acknowledgements

Amin Karbasi was supported by a DARPA Young Faculty Award (D16AP00046) and a AFOSR Young Investigator Award (FA9550-18-1-0160). Ehsan Kazemi was supported by the Swiss National Science Foundation (Early Postdoc.Mobility) under grant number 168574.

References

Bach, F. et al. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends® in Machine Learning*, 6(2-3):145–373, 2013.

Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., and Krause, A. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 671–680. ACM, 2014.

Barbosa, R., Ene, A., Nguyen, H., and Ward, J. The power of randomization: Distributed submodular maximization on massive datasets. In *International Conference on Machine Learning*, pp. 1236–1244, 2015.

Bertsimas, D., Brown, D. B., and Caramanis, C. Theory and applications of robust optimization. *SIAM review*, 53(3): 464–501, 2011.

Beutel, A., Chen, J., Zhao, Z., and Chi, E. H. Data Decisions and Theoretical Implications when Adversarially Learning Fair Representations. *arXiv preprint arXiv:1707.00075*, 2017.

Blake, C. L. and Merz, C. J. Uci repository of machine learning databases [http://www. ics. uci. edu/~ mlearn/mlrepository. html]. irvine, ca: University of california. *Department of Information and Computer Science*, 55, 1998.

Bogunovic, I., Mitrovic, S., Scarlett, J., and Cevher, V. Robust Submodular Maximization: A Non-Uniform Partitioning Approach. pp. 508–516, 2017.

Bogunovic, I., Zhao, J., and Cevher, V. Robust Maximization of Non-Submodular Objectives. In *AISTATS*, volume 84 of *Proceedings of Machine Learning Research*, pp. 890–899. PMLR, 2018.

Borodin, A., Jain, A., Lee, H. C., and Ye, Y. Max-Sum Diversification, Monotone Submodular Functions, and Dynamic Updates. *ACM Transactions on Algorithms (TALG)*, 13(3):41, 2017.

Buchbinder, N., Feldman, M., and Schwartz, R. Online submodular maximization with preemption. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1202–1216. Society for Industrial and Applied Mathematics, 2015.

Chayes, J. T. How Machine Learning Advances Will Improve the Fairness of Algorithms, 2017. URL https://www.huffingtonpost.com/entry/how-to-shorten-a-website-link_us_579bb7aee4b07066ba1ea7dd.

Chekuri, C., Gupta, S., and Quanrud, K. Streaming algorithms for submodular function maximization. In *International Colloquium on Automata, Languages, and Programming*, pp. 318–330. Springer, 2015.

Feldman, M., Harshaw, C., and Karbasi, A. Greed Is Good: Near-Optimal Submodular Maximization via Greedy Optimization. In *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pp. 758–784, Amsterdam, Netherlands, 07–10 Jul 2017. PMLR.

Feldman, M., Karbasi, A., and Kazemi, E. Do Less, Get More: Streaming Submodular Maximization with Subsampling. *CoRR*, abs/1802.07098, 2018. URL http://arxiv.org/abs/1802.07098.

Herbrich, R., Lawrence, N. D., and Seeger, M. Fast sparse Gaussian process methods: The informative vector machine. In *Advances in neural information processing systems*, pp. 625–632, 2003.

Hoerl, A. E. and Kennard, R. W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Kempe, D., Kleinberg, J., and Tardos, É. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146. ACM, 2003.

Krause, A. and Gomes, R. G. Budgeted nonparametric learning from data streams. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 391–398, 2010.

Krause, A. and Guestrin, C. Near-optimal Nonmyopic Value of Information in Graphical Models. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pp. 324–331, 2005.

Krause, A., McMahan, H. B., Guestrin, C., and Gupta, A. Robust submodular observation selection. *Journal of Machine Learning Research*, 9(Dec):2761–2801, 2008.

Kumar, R., Moseley, B., Vassilvitskii, S., and Vattani, A. Fast greedy algorithms in mapreduce and streaming. *ACM Transactions on Parallel Computing*, 2(3):14, 2015.

Lin, H. and Bilmes, J. A Class of Submodular Functions for Document Summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, Stroudsburg, PA, USA, 2011.

Mirrokni, V. and Zadimoghaddam, M. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 153–162. ACM, 2015.

Mirzasoleiman, B., Karbasi, A., Sarkar, R., and Krause, A. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, pp. 2049–2057, 2013.

Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J., and Krause, A. Lazier Than Lazy Greedy. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 1812–1818, 2015.

Mirzasoleiman, B., Karbasi, A., and Krause, A. Deletion-Robust Submodular Maximization: Data Summarization with "the Right to be Forgotten". In *International Conference on Machine Learning*, pp. 2449–2458, 2017.

Mitrovic, S., Bogunovic, I., Norouzi-Fard, A., Tarnawski, J. M., and Cevher, V. Streaming Robust Submodular Maximization: A Partitioned Thresholding Approach. In *Advances in Neural Information Processing Systems*, pp. 4560–4569, 2017.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functionsâĂŤi. *Mathematical Programming*, 14(1): 265–294, 1978.

Orlin, J. B., Schulz, A. S., and Udwani, R. Robust monotone submodular function maximization. In *International Conference on Integer Programming and Combinatorial Optimization*, pp. 312–324. Springer, 2016.

Seeger, M. Greedy forward selection in the informative vector machine. Technical report, Technical report, University of California at Berkeley, 2004.

Singla, A., Tschiatschek, S., and Krause, A. Noisy Submodular Maximization via Adaptive Sampling with Applications to Crowdsourced Image Collection Summarization. In *AAAI*, pp. 2037–2043, 2016.

Smola, A. J. and Schölkopf, B. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

Tzoumas, V., Jadbabaie, A., and Pappas, G. J. Resilient Non-Submodular Maximization over Matroid Constraints. *arXiv preprint arXiv:1804.01013*, 2018.

UberDataset. Uber Pickups in New York City. URL `https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city`.

Wei, K., Iyer, R., and Bilmes, J. Submodularity in data subset selection and active learning. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1954–1963, 2015.

Yue, Y. and Guestrin, C. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems*, pp. 2483–2491, 2011.

Zemel, R., Wu, Y., Swersky, K., Pitassi, T., and Dwork, C. Learning fair representations. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 325–333, 2013.

Zhang, H. The Optimality of Naive Bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA*, pp. 562–567, 2004.