Large-scale Submodular Greedy Exemplar Selection with Structured Similarity Matrices

Dmitry Malioutov
IBM Research
Yorktown Heights, NY

Abhishek Kumar
IBM Research
Yorktown Heights, NY

Ian E.H. Yen
Computer Science Department
University of Texas at Austin

Abstract

Exemplar clustering attempts to find a subset of data-points that summarizes the entire data-set in the sense of minimizing the sum of distances from each point to its closest exemplar. It has many important applications in machine learning including document and video summarization, data compression, scalability of kernel methods and Gaussian processes, active learning and feature selection. A key challenge in the adoption of exemplar clustering to large-scale applications has been the availability of accurate and scalable algorithms. We propose an approach that combines structured similarity matrix representations with submodular greedy maximization that can dramatically increase the scalability of exemplar clustering and still enjoys good approximation guarantees. Exploiting structured similarity matrices within the context of submodular greedy algorithms is by no means trivial, as naive approaches still require computing all the entries of the matrix. We propose a randomized approach based on sampling sign-patterns of columns of the similarity matrix and establish accuracy guarantees. We demonstrate significant computational speed-ups while still achieving highly accurate solutions, and solve problems with up-to millions of data-points in around a minute or less on a single commodity computer.

1 Introduction

With the prevalence of large-scale datasets coming from social media, computational biology, finance and engineering it has been difficult to apply some of the more computationally intensive machine learning approaches such as probabilistic Graphical models, kernel methods and Gaussian processes. Simple tools like logistic regression trained via stochastic gradient descent are instead prevalent in web companies that need to deal with hundreds of millions and more samples [16]. Exemplar clustering suggests a path to extend some of the powerful ML methods to such domains by finding small representative subsets of data to summarize the large dataset, and apply the method on the summary. Other uses of exemplar clustering include summarizing data for humans (document and video summarization [14]), facility location in OR, and active learning [7] to prioritize the annotation in large unlabeled datasets where annotating the entire dataset is out of the question.

An integer programming formulation for exemplar clustering is itself intractable, being an NP-hard problem [21]. Hence, a variety of approximate schemes have been developed including LP relaxations [3], message-passing algorithms [8, 13], and heuristic algorithms such as Partitioning around Medoids (PAM) [11]. The most successful methods balancing scalability and good accuracy guarantees[1] have been based on the theory of submodular maximization [12]. Exemplar clustering (with some mild assumptions) can be shown to be submodular, i.e. to satisfy diminishing marginal gains. Thus, invoking the celebrated result of [21], the greedy approach that adds exemplars in order of their marginal gains has an $1 - 1/e$ approximation guarantee. A lazy evaluation approach called "Lazy Greedy" further exploits the diminishing marginal gains property to accelerate the solution [18].

Nevertheless, for very large scale data-sets even greedy algorithms become computationally infeasible, as the exemplar clustering cost-function is non-local, and evaluating the marginal gains for each exemplar candidate requires evaluating the distance to each other point in the dataset. A number of directions have been considered to speed up basic greedy (and lazy greedy) methods. Simple surrogate functions (such as modular bounds) approximating the submodular function have been proposed in [26]. The paper also introduced a nearest neighbors pruning approach

[1] There is a rich theory on LP relaxations for exemplar clustering, but the methods are less scalable, having to solve linear programs with $O(n^2)$ variables, where n is the number of data-points.

to reduce the set of candidates to consider. This idea has been further explored in [15]. A distributed algorithm that randomly splits the data among several machines, selects exemplars from each subset, and then combines them was proposed in [20]. This however requires further approximation by a factor of $1/min(k, m)$ where k is the number of exemplars, and m is the number of machines. A stochastic sub-sampling idea called "Lazier than lazy greedy" was proposed in [19] which takes a small random subset of potential exemplars into consideration at each stage. By taking the sample size proportional to $n/k \log(1/\epsilon)$ the running time of the algorithm executed for k iterations becomes independent of k, and is still able to achieve an $1 - 1/e - \epsilon$ approximation guarantee (in expectation). This result was also applied in the streaming setting [2].

We pursue a different approach where we assume that the similarity matrices are either exactly represented as certain structured matrices (such as low-rank, sparse plus low-rank, product of sparse matrices, Toeplitz, e.t.c), or can be approximated by such. For example, using the popular word2vec representation for words and sentences [17] along with cosine similarity gives rise to an explicit low-rank similarity matrix. Squared Euclidean distance matrices (and corresponding similarity matrices) are also exactly low-rank for low-dimensional feature spaces. While simply using the structured matrix for the greedy algorithm reduces its memory footprint, it still requires quadratic computation in number of samples n. We show that the bottleneck of the problem involves a low-rank positive column sum problem. We develop a randomized approximation technique based on sampling column sign-patterns and establish theoretical guarantees which can reduce the computation to linear in n. The proposed approach is guaranteed to generate a better solution than the stochastic sampling approach [19] at the cost of a very modest increase in computation. We consider numerical studies on a number of large-scale machine learning problems and demonstrate dramatic increase in speed compared to plain greedy and lazy-greedy methods with only a minor corresponding loss in objective values.

The outline of the paper is as follows. We first overview submodular maximization, and prior work on exact and approximate greedy algorithms in Section 2. We discuss structured approximation matrices and our approach for approximate greedy optimization in Section 3. We present theoretical guarantees in Section 4 and experimental results in Section 5.

2 Background: submodular optimization

In this paper we consider an approach for exemplar clustering based on *submodular maximization*. We start by reviewing the notions of submodularity, greedy algorithms and their guarantees, and the recent extensions.

Let $\mathcal{V}$ be a set of size n. A set function $f : 2^{\mathcal{V}} \to \mathbb{R}$ is called submodular if it satisfies a diminishing marginal gains property: if $A \subseteq B$, and $a \in \mathcal{V}$ then

$$f(A \cup a) - f(A) \geq f(B \cup a) - f(B)$$

Intuitively, there is more benefit (higher marginal gain) to add a new element a to the smaller set A than to its superset B. If for all $A \subseteq B$ it holds that $f(A) \leq f(B)$ then the function is called *monotone submodular*. If $f(\varnothing) = 0$ then the function is called homogeneous. A celebrated result by Neumhauser et. al establishes that a simple greedy algorithm achieves an $1 - \frac{1}{e}$ approximation to the optimal objective to non-negative cardinality constrained monotone submodular maximization [21]. The greedy algorithm starts with an empty set, and at each stage adds the element which has the maximum marginal gain $\Delta f(a|A) = f(A \cup a) - f(A)$ to the current selected set A.

The exemplar clustering problem that we consider in the paper has the following form. We have a collection $\mathcal{V}$ of data-points,[2] and a similarity function $s : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$. We would like to faithfully summarize the entire data-set (maximize the pairwise similarity between each point and its exemplar) using at most k exemplars:

$$\max_{A \subset \mathcal{V}} \sum_{i \in \mathcal{V}} \max_{j \in A} s(i, j) \text{ where } |S| \leq k \tag{1}$$

One can verify that the exemplar clustering problem above corresponds to nonnegative monotone submodular maximization with a cardinality constraint, and hence the greedy algorithm can be applied with the corresponding approximation guarantee. The greedy algorithm can be sped up further using the same diminishing returns property of marginal gains [18] by lazy evaluation, by keeping the elements in a priority queue sorted according to marginal gain.

Unfortunately, the evaluation of the marginal gain for a single candidate exemplar requires $O(n)$ computation, and we consider $O(n)$ candidates at each stage. Hence the greedy algorithm still does not scale to very large data-sets beyond a few hundred thousand data points. Improving the scalability of the greedy algorithm for exemplar clustering has been an active research area and a number of approximation strategies have been proposed at the cost of worse approximation guarantees. The existing proposals include stochastic greedy, distributed algorithms and nearest-neighbor graph approximations that we have described in the introduction. Next we describe how we can exploit structured similarity matrices and a randomized ap-

[2] All our results also extend to the non-symmetric version of the problem where the data-points come from set $\mathcal{V}_1$ and exemplars have to be selected from $\mathcal{V}_2$, for example the facility location problem in Operations Research. Also, allowing rewards $h_i \geq 0$ for making data-point i an exemplar, still keeps the problem monotone and submodular.

proach based on sampling sign-patterns of columns to propose an elegant scalable solution for exemplar clustering with good approximation guarantees.

3 Greedy exemplar selection with low-rank similarity matrices

We consider the greedy approach for exemplar clustering. As inputs we have the similarity matrix S and a budget k. At each stage t we have the current active set $\mathcal{A}^t$ of exemplars chosen at the previous iteration. We would like to evaluate the marginal gain of new elements $a \in \mathcal{V} \setminus \mathcal{A}^t$.

Suppose that at step t we have computed the optimal assignment $\hat{j}^t(i) = \arg\max_{j \in \mathcal{A}^t} s(i,j)$ for each datapoint i. We will refer to the attained objective as $z_i = s(i, \hat{j}^t(i))$. At iteration $t+1$, we compute the marginal gain for each $a \in \mathcal{V} \setminus \mathcal{A}^t$. The closest element to i in $\mathcal{A}^t \cup a$ either remains the same as in the previous step, or it becomes a. Hence the marginal gain $\Delta f(a|A)$ of adding a to $\mathcal{A}^t$ is

$$\sum_i \max\Big(s(i,a),\ s(i,\hat{j}^t(i))\Big) - s(i,\hat{j}^t(i)) \qquad (2)$$

Define $[X]_+ = \max(X, 0)$. Then the computational bottleneck is computing the vector of marginal gains, i.e. the maximum positive column sum of the matrix $R = S - \mathbf{z}\mathbf{1}^T$:

$$\max_j \sum_i [R_{ij}]_+ = \max_j \sum_i [S_{ij} - z_i]_+ \qquad (3)$$

Naive implementation requires computing all the elements of this matrix[3]. We show that if the matrix S has a structured representation and allows fast matrix action (mutliplication of the matrix by a vector) – then we can exploit a randomized algorithm with good guarantees.

We first analyze the case of low-rank matrices, $S = \tilde{U}\tilde{V}^T$, where $\tilde{U}$ and $\tilde{V}$ are $n \times d$. We show that we can use a number of other structured matrices in Section 3.4. Now, if S is low-rank with rank d then R is also low-rank with rank at most $d+1$. Let $U = [\tilde{U},\ -\mathbf{z}]$ and $V = [\tilde{V},\ \mathbf{1}]$. Then $R = UV^T = S - \mathbf{z}\mathbf{1}^T$. Note that computing the plain column sum for low-rank matrices is trivial: $\mathbf{1}^T R = (\mathbf{1}^T U)V^T$ thus reducing the computational complexity from $O(n^2)$ to $O(nd)$. However, the positive sign in $\mathbf{1}^T[R]_+$ makes it considerably more difficult to exploit the low-rank factorization. We study this problem next.

3.1 Maximum positive column sum problem

We would like to efficiently compute the maximum positive column sum for a low-rank matrix $R = UV^T$:

$$\max_j \sum_i [R_{ij}]_+ = \max_j \sum_i \left([UV^T]_+\right)_{ij} \qquad (4)$$

Algorithm 1 Approximate max-positive column sum

```
1. Uniformly at random sample $r$ columns from the matrix $R$
and compute their sign patterns $\mathbf{q}_i$.
2. Aggregate sign-patterns into matrix $Q = [\mathbf{q}_1, ...\mathbf{q}_r]$.
3. Let $Y = (Q^T U)V$ and let $\mathbf{y} = \max_i Y_{ij}$.
4. Find $\hat{j} = \arg\max_j \mathbf{y}_j$.
```

Suppose we knew the binary ($\{0,1\}$) sign-pattern $\hat{\mathbf{q}}$ of the column achieving the maximum in (4), i.e. $q_i = 1_{\{R_{i\hat{j}} > 0\}}$ where $\hat{j}$ is the column achieving the $\arg\max$ in (4). Then we could compute

$$\max_j \sum_i [R_{ij}]_+ = \max_j \sum_i \left([UV^T]_+\right)_{ij} = \max_j [(\hat{\mathbf{q}}^T U)V^T]_j \qquad (5)$$

Also note, that for an arbitrary sign-pattern $\mathbf{q}$, we have

$$\max_j [(\hat{\mathbf{q}}^T U)V^T]_j \geq \max_j [(\mathbf{q}^T U)V^T]_j \qquad (6)$$

We suggest to take a collection of r random sign-patterns $\{\mathbf{q}_1, ..., \mathbf{q}_r\}$ and evaluate the maximum value of the r.h.s. in (6) over all of them. We take the sign-patterns from a subset of columns of R sampled uniformly at random (another choice can be i.i.d. binary vectors). Surprisingly, such a simple scheme can be shown to satisfy good approximation guarantees, which we describe in Section 4, and also to give very accurate numerical results. In particular, it is guaranteed to produce better results than the stochastic greedy approach [19] with the same subset of columns.

The procedure is described in Algorithm 1. It produces a lower bound $\sum_i R_{i\hat{j}}$ on the maximum positive column sum of R:

$$\hat{j} := \arg\max_j \max_{\tilde{j} \in \{j(1), ..., j(r)\}} \mathbf{q}_{\tilde{j}}^T R_{:,j}. \qquad (7)$$

3.2 Scalable submodular greedy algorithm

Now, to use this algorithm for efficient greedy submodular optimization, we have to repeat it for k rounds of greedy to generate k exemplars. At each round we simply recompute the vector $\mathbf{z}$ and $R = [\tilde{U},\ -\mathbf{z}][\tilde{V},\ \mathbf{1}]^\top$ and invoke Algorithm 1. At the initial stage the vector $\mathbf{z}$ is zero, and the initial marginal gains are simply the column-sums of S. We summarize the steps in Algoritm 2.

The greedy algorithm is recovered if we find the exact column maximizing marginal gains: $\arg\max_j \sum_i ([R]_+)_{ij}$. This however would require $O(n^2)$ computation. By using the approach from Section 3.1 with r random signs, it can be reduced to $O(nr)$ by allowing approximate answers.

3.3 Further speed-ups

The approach proposed in the previous section can dramatically reduce the computation of the greedy method from

[3] Lazy greedy algorithms avoid computing some elements but still evaluate a large number of columns of this matrix.

Algorithm 2 Low Rank GReedy (LRGR)

```
Input: $S = \tilde{U}\tilde{V}^T$
Initialize: $\hat{j}^1 = \arg\max_j \sum_i [\tilde{U}\tilde{V}^T]_{ij}$. Set $\mathcal{A}^1 = \{\hat{j}^1\}$.
Evaluate $z_i = s(i, \hat{j}^1)$. Set $t = 1$.
for $i = 1 \dots r$ do
    i. Advance $t \rightarrow t + 1$
    ii. Let $R = \tilde{U}\tilde{V}^T - \mathbf{z}1^T$.
    iii. Use (7) to approximately find
        $\hat{j}^t \approx \arg\max_j \sum_i ([R]_+)_{ij}$.
    iv. $\mathcal{A}^t = \mathcal{A}^{t-1} \cup \hat{j}^t$
    v. Update $\mathbf{z}^t$: $z_i^{t+1} = \max_{j \in A^t} s(i, j)$
end for
```

$O(n^2)$ to $O(nr)$ per each round of the greedy method. Yet, we can improve the running time and accuracy further.

In the previous section the matrix Q of sign-patterns was recomputed from scratch at each greedy iteration. Note that if we store this matrix from the previous iteration, then we only need to update $Q^T\mathbf{z}$, while $Q^T\tilde{U}$ remains the same. While reusing the same matrix Q on all iterations may significantly impact accuracy, we can keep generating new sign patterns but also keep re-using the computationally inexpensive pre-computed old-sign patterns.

We also note that we can compute the complement sign-patterns very inexpensively. Let $\bar{\mathbf{q}} = \mathbf{1} - \mathbf{q}$. Then $(\bar{\mathbf{q}}^T U)V^T = (\mathbf{1}^T U)V^T - (\mathbf{q}^T U)V^T$. Thus we can add another r complement sign-patterns at the cost of computing the column sum of $R = UV^T$.

Finally, the algorithm proposed in the previous section aims to speed up the computation of the plain greedy algorithm. In [19] an extension of stochastic greedy was proposed to also accelerate the lazy-greedy approach by caching previously computed marginal gains and storing them in a priority queue. The same ideas can be used with our randomized sign-pattern approach to develop its lazy-greedy variant.

3.4 Low-rank and other structured similarity matrices

In Sections 3.1 and 3.2 we assumed that the similarity matrix S is low-rank, which can happen if we use inner-product similarity measure in a low-dimensional vector space (e.g word2vec). Note that the matrix of pairwise squared Euclidean distances is also low-rank for low-dimensional feature spaces, and this also holds for matrices based on arbitrary Bregman divergences, and hence for the corresponding similarity matrices.

In addition to the low-rank assumption, we can also use a variety of other structured matrix factorizations in exactly the same way, as long as they allow fast-matrix action. Suppose $S = \tilde{U}\tilde{V}^T$, then to apply the approach in Section 3.1 we simply need to be able to quickly compute $Q\tilde{U}$ and $(Q\tilde{U})\tilde{V}^T$. For example if $\tilde{U}$ and $\tilde{V}$ have high-dimensional but very sparse rows (e.g. the sparse one-hot bag-of-words representation for text), then the matrix-vector products depend on the number of non-zero entries in $\tilde{U}$ and $\tilde{V}$.

Other factorizations allowing fast matrix action include sparse plus low-rank matrices, Fourier matrices, circulant and convolution matrices, and even more advanced matrix approximations including block-diagonal low-rank matrices [24] and matrices with low displacement operators [25].

4 Theoretical analysis

In this section we show that LRGR achieves an approximately optimal solution in expectation when sufficiently many sign patterns are sampled. In the first part we show approximation bounds that do not assume any structure on the similarity matrix S. Subsequently, we try to provide intuition behind how the low-rank structure on S can help in achieving better solution by analyzing the case of rank-2 similarity matrix. Finally, we analyze the case of data distributed across tight clusters.

4.1 Structure-oblivious approximation guarantees

The results in this section do not exploit the structure on the similarity matrix. The proof proceeds along similar lines as [19]. We prove the result for the symmetric case when the candidate exemplars belong to the set of points. Let $\mathcal{A}^*$ be the optimal solution of problem (1) with $|\mathcal{A}^*| = k$ (if the algorithm terminates with $|\mathcal{A}^*| < k$, it will return the optimal solution [21]), and $\mathcal{A}^t$ be the solution of LRGR at step t. Let $\mathcal{V}$ denote the full index set of all the points ($\mathcal{V} = \{1, 2, \dots, n\}$). Let $f(\mathcal{A}^t)$ be the objective value for set $\mathcal{A}^t$ and $\Delta(a|\mathcal{A}^t) := f(\mathcal{A}^t \cup a) - f(\mathcal{A}^t)$.

Using submodularity, we have

$$f(\mathcal{A}^*) - f(\mathcal{A}^t) \leq \sum_{a \in \mathcal{A}^* \setminus \mathcal{A}^t} \Delta(a|\mathcal{A}^t) \qquad (8)$$

The following Lemma bounds the expected potential gain of LRGR after step t.

Lemma 4.1. *Given a current solution $\mathcal{A}^t$, the expected gain of LRGR in the next step is at least $\frac{1-e^{-kr/n}}{k} \sum_{a \in \mathcal{A}^* \setminus \mathcal{A}^t} \Delta(a|\mathcal{A}^t)$ when signs from r randomly picked columns are used.*

Proof. Let $\mathcal{R}$ be the set of randomly picked columns with $|\mathcal{R}| = r$. The probability that one of the columns from $\mathcal{A}^* \setminus \mathcal{A}^t$ is picked, is given by

$$\begin{aligned} P[\mathcal{R} \cap (\mathcal{A}^* \setminus \mathcal{A}^t) \neq \phi] &= 1 - P[R \cap (\mathcal{A}^* \setminus \mathcal{A}^t) = \phi] \\ &= 1 - \left(1 - \frac{|\mathcal{A}^* \setminus \mathcal{A}^t|}{|\mathcal{V} \setminus \mathcal{A}^t|}\right)^r \\ &\geq 1 - e^{-\frac{r|\mathcal{A}^* \setminus \mathcal{A}^t|}{|\mathcal{V} \setminus \mathcal{A}^t|}} \geq 1 - e^{-\frac{r|\mathcal{A}^* \setminus \mathcal{A}^t|}{n}} \end{aligned}$$

Since $0 \le |\mathcal{A}^* \setminus \mathcal{A}^t| \le k$, using Jensen's inequality we have

$$1 - e^{-\frac{r|\mathcal{A}^*\setminus\mathcal{A}^t|}{n}} \ge \left(1 - e^{-\frac{rk}{n}}\right)\frac{|\mathcal{A}^* \setminus \mathcal{A}^t|}{k}$$

If LRGR picks one of the columns from $\mathcal{A}^* \setminus \mathcal{A}^t$, the score for that column will be evaluated exactly. Scores for the columns whose sign pattern matches that of the picked column will also be evaluated exactly. The scores for all other columns will be underestimated. As LRGR takes a maximum over all the scores, it will add a column to $\mathcal{A}^t$ which will give a gain equal to at least that of the picked column from $\mathcal{A}^* \setminus \mathcal{A}^t$. As any of the columns from $\mathcal{A}^* \setminus \mathcal{A}^t$ is equally likely to be picked, the exemplar a^{t+1} added by LRGR will yield

$$\begin{aligned}\mathbf{E}[\Delta(a^{t+1}|\mathcal{A}^t)] &\ge P[\mathcal{R} \cap (\mathcal{A}^* \setminus \mathcal{A}^t) \neq \phi]\frac{\sum_{a\in\mathcal{A}^*\setminus\mathcal{A}^t}\Delta(a|\mathcal{A})}{|\mathcal{A}^* \setminus \mathcal{A}^t|}\\ &\ge \frac{1-e^{-kr/n}}{k}\sum_{a\in\mathcal{A}^*\setminus\mathcal{A}^t}\Delta(a|\mathcal{A}^t).\end{aligned}$$

□

Combining Lemma 4.1 with Eq. 8 we get

$$f(\mathcal{A}^*) - f(\mathcal{A}^t) \le \frac{k}{1-e^{-kr/n}}\mathbf{E}[f(\mathcal{A}^{t+1}) - f(\mathcal{A}^t)] \quad (9)$$

Taking expectation over $\mathcal{A}^t$, we have

$$\mathbf{E}[f(\mathcal{A}^{t+1}) - f(\mathcal{A}^t)] \ge \frac{1-e^{-kr/n}}{k}\mathbf{E}[f(\mathcal{A}^*) - f(\mathcal{A}^t)] \quad (10)$$

Unrolling the above by induction, we get

$$\begin{aligned}\mathbf{E}[f(\mathcal{A}^k)] &\ge \left(1 - \left(1 - \frac{1-e^{-kr/n}}{k}\right)^k\right)f(\mathcal{A}^*)\\ &\ge \left(1 - e^{-(1-e^{-kr/n})}\right)f(\mathcal{A}^*)\\ &\ge \left(1 - e^{-1}(1+e^{-kr/n})\right)f(\mathcal{A}^*)\\ &= (1 - 1/e - e^{-kr/n})f(\mathcal{A}^*).\end{aligned}$$

Note that the above bound turns vacuous for $r \le \frac{n}{k}\ln\frac{1}{1-1/e} \approx 0.459\frac{n}{k}$. When the similarity matrix is allowed to have negative entries, the submodular objective function may lose nonnegativity but it still remains monotonic. In this case, one can obtain a guarantee of the form:

$$f(\mathcal{A}^*) - \mathbf{E}[f(\mathcal{A}^k)] \le (e^{\frac{1-k}{k}} + e^{\frac{-kr}{n}})(f(\mathcal{A}^*) - \mathbf{E}[f(\mathcal{A}^1)]).$$

4.2 Approximation guarantee under low-rank assumption

The analysis in the previous section did not use the low-rank assumption on the matrix R. We show that this assumption is helpful not only to improve the computational complexity of the proposed algorithm, but also to improve the approximation compared to the stochastic greedy approach. To build some intuition of why the low-rank assumption is helpful, consider the maximum possible number of distinct sign-patterns of columns of the matrix R. If the matrix is full-dimensional, then any sign-pattern is possible, and there are 2^n such sign patterns out of which n are chosen. However, if $R = UV^T$, and U and V are $n \times d$, then for a fixed U, the number of possible sign-patterns of columns of R is reduced. In particular, as we show below, for the 2-dimensional case, the number of possible sign-patterns is $2n$ (instead of 2^n): U has two columns, and the sign patterns of any linear combinations of these two columns generate only a small subset of possible sign patterns.

More generally, if U is d-dimensional, then using the results in [10] for number of regions created by n hyperplanes (in general position) in d dimensions, we have that the number of possible sign-patterns of R for a fixed U is given by:

$$2\sum_{i=1}^{d}\binom{n-1}{i-1} \quad (11)$$

For the 2-dimensional case, this number is $2n$. If we assumed that the column sign-patterns in our matrix R are sampled uniformly at random with replacement from the $2n$ possible choices, then the expected number of unique sign-patterns is $\sum_{i=0}^{n-1}(\frac{2n-1}{2n})^i = 2n(1-(1-1/(2n))^n) \approx 0.787n$. So we expect some number of columns with repeated sign-patterns in the low-dimensional case. This makes it easier for us to get a sign-pattern from our r sampled columns of R that matches the sign-pattern of one of the exemplars.

Let $\mathcal{A}^*$ be the index set of optimal k exemplars. For the 2-dimensional case, we consider the probability that at least one of the r randomly sampled columns, indexed by set $\mathcal{R}$, will completely agree in signs with at least one of the exemplars. The worst case for this probability is when all the exemplars have exactly the same sign pattern since we reduce our set of candidate signs that the sampled columns can match to. This worst case probability is given as:

$$\begin{aligned}&P(\mathcal{R} \cap A^* \neq \phi) + P(\mathcal{R} \cap A^* = \phi)(1 - ((2n-1)/2n)^r)\\ &= 1 - (1 - k/n)^r + (1 - k/n)^r(1 - ((2n-1)/2n)^r)\\ &\ge 1 - e^{-kr/n}e^{-r/(2n)}.\end{aligned}$$

The first term in the first line corresponds to the probability of selecting one of the r sampled columns to be one of the exemplars, and the second term corresponds to picking a column that misses the set of exemplars, but whose sign-pattern agrees with the sign pattern of one of the exemplars. This probability of at least one agreement in sign patterns is also a lower bound on the probability of Algorithm 1 recovering a column whose score is at least that of one of the exemplars. For the best case when all the exemplars differ with each other in at least one place in sign patterns, the lower bound on the recovery probability increases to $(1 - e^{-kr/n}e^{-kr/(2n)})$. If we ignore the structure on R, the lower bound loosens to $(1 - e^{-kr/n})$ which was used in Section 4.1.

We should also note that this bound, although better than stochastic greedy [19], is based on several worst-case scenarios and is still highly pessimistic for practical settings. The uniform sampling assumption of sign patterns implies that all regions created by the n hyperplanes [10] are equally likely to contribute to the sign patterns, whereas in practice the data is often non-uniformly distributed and even occurs in clusters.

Furthermore, the analysis does not consider inexact matches, where the sampled columns differ in sign patterns with the exemplars only at a few places. This disagreement in a small number of places will result in underestimating the positive sum of the exemplar column to some extent. However, if the score of the exemplar is mainly contributed by a few data points (i.e., a few rows) and the gap between exemplar score and the best non-exemplar score is sufficient, the algorithm may still have a good chance in picking the exemplar column in next iteration. If this gap is low, we do not lose much in terms of objective value by picking one of the runner-up columns. Indeed, we empirically observe a considerable performance gain (in terms of objective value) over stochastic greedy on several real datasets in Section 5.

4.3 Approximation guarantees under cluster assumption

Here we consider another simplified scenario for better understanding of the sampling approach (7). We assume data points form K disjoint clusters $\mathcal{C}_1$, $\mathcal{C}_2$, ..., $\mathcal{C}_K$ satisfying $0 < m \leq |\mathcal{C}_k| \leq M \leq n$ and

(a) $S_{ij} \geq \kappa, \forall i, j \in \mathcal{C}_k$ for some constant $\kappa > \frac{4}{4+m/M}$.

(b) $S_{ij} \in [c-\epsilon/2, c+\epsilon/2]$ for i, j not in the same cluster, where c, ϵ are constants satisfying $c < (\kappa - \epsilon)/2$ and $\epsilon < \frac{m}{4n}\kappa$.

The cluster assumption ensures that points in the same cluster $\mathcal{C}_k$ have similarity at least κ, while points of different clusters has similarity of bounded variance (in range $[c - \epsilon/2, c + \epsilon/2]$). Note the low-variance condition for points of different clusters is often satisfied in problem of higher dimension due to the *curse of dimensionality*, where uncorrelated points can have maximum distance indiscernible to the minimum distance [4]. For example, in document categorization, the semantically-unrelated documents have only *stop words* in common, which causes unrelated documents to have almost the same similarity to each others. Under the cluster assumptions (a)-(b), we show that the sampling estimator (7) has approximation error bounded by $2n\epsilon$ as follows. A related clustering assumption was used to analyze convex relaxations for exemplar clustering in [6].

Theorem 4.1. *Let j^* and $\hat{j}$ denote the greedy column selected by exact criteria* (3) *and sampling criteria* (7) *with r samples respectively. For data satisfying clustering assumption (a)-(b), we have*

$$\sum_{i=1}^{n}[R_{i\hat{j}}]_+ \geq \sum_{i=1}^{n}[R_{ij^*}]_+ - 2n\epsilon \tag{12}$$

holds for each iteration $t \leq K$ with probability at least $1 - \delta$ if

$$r \geq \frac{n}{|\mathcal{C}(j^*)|}\ln(\frac{1}{\delta}). \tag{13}$$

where $|\mathcal{C}(j^*)|$ is the size of the cluster that contains j^*.

Proof. For $t = 1$, (12) holds directly since $R = S$ and no sampling is used (so $\hat{j} = j^*$). For $t > 1$, $\mathcal{C}(j^*)$ must be a cluster that does not contain any exemplar chosen in previous iterations. Otherwise, suppose $\mathcal{C}(j^*)$ contains an exemplar j' selected in previous iterate. We have

$$\begin{aligned}\sum_{i=1}^{n}[R_{ij^*}]_+ &\leq \sum_{i\in\mathcal{C}(j^*)}[S_{ij^*} - S_{ij'}]_+ + \sum_{i\notin\mathcal{C}}[R_{ij^*}]_+ \\ &\leq |\mathcal{C}(j^*)|(1-\kappa) + n\epsilon \\ &< \frac{m\kappa}{4} + \frac{m\kappa}{4} = m\kappa/2.\end{aligned}$$

by assumption (a)-(b). However, picking any point j from a cluster $\mathcal{C}$ that does not contain exemplar from previous iterates gives

$$\begin{aligned}\sum_{i=1}^{n}[R_{ij}]_+ &\geq \sum_{i\in\mathcal{C}(j^*)}[S_{ij^*} - (c+\frac{\epsilon}{2})]_+ \\ &\geq \sum_{i\in\mathcal{C}(j^*)}[\kappa - \kappa/2]_+ \geq m\kappa/2,\end{aligned}$$

which leads to contradiction.

Given that $\mathcal{C}(j^*)$ is a cluster not containing exemplar selected from previous iterates, consider $\hat{j}$ selected by the sampling estimator (7). With number of samples r satisfying (13), we have at least $1-\delta$ probability that one of the sign-pattern vectors $\mathbf{q}_{\tilde{j}}$ is produced by a point $\tilde{j} \in \mathcal{C}(j^*)$, which has

$$q_{i\tilde{j}} := 1[S_{i\hat{j}} - z_i] = 1 \text{ , } \forall i \in \mathcal{C}(j^*) \tag{14}$$

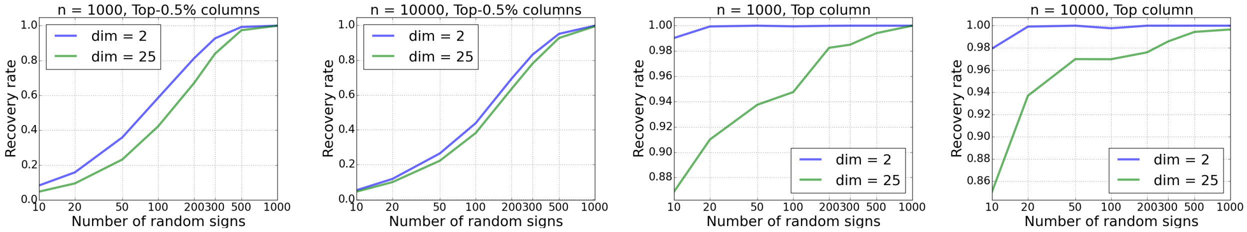

Figure 1: Average recovery rates vs. number of sampled columns (log-scale) for Algorithm 1 over 100 random trials. Data is generated from uniform distribution on the unit sphere (left two plots) and standard Cauchy distribution (right two plots).

since $z_i = S_{ij}$ for some $j \notin \mathcal{C}(j^*)$. Therefore, we have

$$\begin{aligned}\sum_{i=1}^{n}[R_{i\hat{j}}]_+ &\geq \max_{j\in\{j(1),..,j(r)\}} \sum_{i=1}^{n} q_{ij} R_{i\hat{j}} \\ &\geq \max_{j\in\{j(1),..,j(r)\}} \sum_{i=1}^{n} q_{ij} R_{ij^*} \geq \sum_{i=1}^{n} q_{i\tilde{j}} R_{ij^*} \\ &\geq \sum_{i\in\mathcal{C}(j^*)} q_{i\tilde{j}} R_{ij^*} + \sum_{i\notin\mathcal{C}(j^*)} q_{i\tilde{j}} R_{ij^*} \\ &\geq \sum_{i\in\mathcal{C}(j^*)} [R_{ij^*}]_+ - n\epsilon,\end{aligned} \tag{15}$$

where the last inequality is due to (14), assumption (b) and $R_{ij^*} := S_{ij^*} - S_{ij} > 0$ for some $j \notin \mathcal{C}(j^*)$. On the other hand,

$$\begin{aligned}\sum_{i=1}^{n}[R_{ij^*}]_+ &= \sum_{i\in\mathcal{C}(j^*)} [R_{ij^*}]_+ + \sum_{i\notin\mathcal{C}(j^*)} [R_{ij^*}]_+ \\ &\leq \sum_{i\in\mathcal{C}(j^*)} [R_{ij^*}]_+ + n\epsilon.\end{aligned} \tag{16}$$

Combining (15) and (16) leads to the conclusion (12). □

5 Experimental results

We now study the performance of the proposed approach on numerical experiments. We first consider the low-rank maximum positive column sum problem from Section 3.1 corresponding to one stage of the greedy algorithm, and then the low-rank greedy algorithm from Section 3.2.

5.1 Synthetic experiments

In this section we perform numerical simulations to study the behavior of the proposed sign sampling approach in Section 3.1 in recovering the top scoring columns. In the first experiment, we independently sample rows of $U \in \mathbb{R}^{n\times d}$ and $V \in \mathbb{R}^{n\times d}$ from a uniform distribution on the unit sphere $\mathbb{S}^{d-1}$, and take the similarity matrix to be $S = UV^\top$. We conduct 100 random trials of generating pairs of U and V. For every sampled pair U and V, we randomly sample 100 different sets of sign patterns and use Algorithm 1 to estimate the highest scoring column using each set. We report the fraction of times the estimated column is among the top scoring 0.5% columns over all these random trials. Figure 1 shows the recovery rate versus the number of sign patterns for $n = 1K, 10K$ and $d = 2, 25$ which clearly increases with the number of sampled columns.

We repeat the same experiment with U and V generated according to the heavy-tailed Cauchy distribution, and we report the much more stringent recovery rate of the top scoring column this time. We observe that Algorithm 1 performs exceptionally well in recovering the top scoring column for both $d = 2$ and 25. Intuitively, this happens due to the heavy tailed nature of Cauchy distribution, where only a small subset of rows of S will contain the maximums and minimums for each column. Hence, getting the correct sign-pattern for these few rows will greatly increase the probability of recovering the top column by Algorithm 1. This example highlights the fact that the proposed LRGR is capable of taking advantage of the underlying distribution whereas stochastic greedy approach [19] remains oblivious to the distribution of entries of S since it computes the maximum only over the sampled subset of columns. In the Cauchy example, stochastic greedy will have a recovery probability of ~ 0.01 for $n = 10K, r = 100$ irrespective of d, whereas Algorithm 1 empirically shows a recovery rate of ~ 0.96 for $d = 25$.

5.2 Experiments on Machine learning datasets

We compare hte proposed randomized low-rank greedy approach (LRGR) to several other competing methods for approximate exemplar clustering, focusing on alternative approximate greedy methods. We have not considered integer programming and message passing based methods as they do not scale to the dataset sizes we consider in the experiments. We describe the contestants first[4]. We consider both data-sets with low-dimensional feature representations, and data-sets with high-dimensional but sparse feature representation. For simplicity we focus on the inner-product similarity measure here. The experiments are run on a sin-

[4] All the algorithms are written in python relying on *numpy* and *scipy* packages for dense and sparse linear algebra.

gle commodity computer with 16 Gb of RAM.

Plain and lazy greedy As a baseline we used the standard greedy and lazy greedy algorithms, which do use the low-rank structure to reduce the memory footprint, but which compute all the required entries of the similarity matrix. We also use the efficient update for the t-th step of the greedy method, where we do not recompute the optimal assignment of points to exemplars from scratch, but we reuse the previous assignment at the $t-1$-st step – and only compare its best assignment to the new added exemplar. Lazy greedy achieves the same objective value as plain greedy. In our experiments with small number of exemplars the running time of lazy-greedy was similar or slower than plain greedy, so the time is not reported. Lazy-greedy can become significantly faster than plain greedy when the number of exemplars is much higher, and it typically accelerates (benefits more from lazy evaluation) at later stages.

Random subset As a very simple baseline we obtain a random subset of points as exemplars, and compute the cost of optimal assignments of points to these exemplars.

K-means with cluster medoids (KmeansCTR). K-means algorithm produces cluster centers that are not data-points but their convex combinations. We use a heuristic that ignores k-means cluster centers, and instead computes the centroid (1-medoid) for each of the k-clusters independently. Initialization is done using k-means++ [1]. Note that for similarity matrices other than cosine similarity comparison with k-means will not be meaningful, as k-means assumes Euclidean distance between data-points.

Stochastic greedy (StochGr) The approach [19] selects at each step a small subset of columns of the similarity matrix, and only searches for exemplars among this set. We use the same subset-size 100 as for the proposed algorithm.

KNN-greedy (KnnGr) A K-nearest neighbor graph for the data is used to create a sparse approximation to the similarity matrix which is non-zero for the K nearest neighbors only [15, 26]. We use 100 nearest neighbors to give the same number of parameters as stochastic greedy and the proposed algorithm. We report two times for this algorithm: total time taken including the construction of the KNN-graph (pre-processing step needed to run the algorithm), and just the time for exemplar clustering starting with the sparse KNN graph already provided.

LR-greedy (LRGR) This is our proposed algorithm which uses sign-patterns of 100 random columns selected at each iteration from the similarity matrix.

Datasets. We use a collection of standard machine learning datasets from the UCI archive [5] of various sizes and numbers of features (both dense and sparse), and report the time and approximation quality in the table below. The data-sets with label (RF) use a Random-Feature Kernel Approximation method [23] to perform exemplar clustering with (RBF-Laplacian) Kernelized features, namely the Random Binning Features.

The final data-set is the World TSP data set which contains the latitude and longitude of 1,904,711 cities in the world[5]. We use geographical distance (computed via polar coordinate flat-Earth formula) between pairs of cities. To obtain a low-rank approximation to the distance matrix $\mathbb{D}$, we sampled $100n$ pairs of distances and use low-rank matrix completion[6] to find $UV^T \approx \mathbb{D}$. Using rank $d = 20$, we obtain a decomposition UV^T with testing RMSE $< 10^{-2}$. We use the corresponding similarity matrix $S = D_{\max} - UV^T$, where $D_{\max}$ is the maximum element of $\mathbb{D}$. Note that due to the non-Euclidean nature of the similarity matrix, comparison with Kmeans-CTR is not meaningful.

Results We report the objective values in table 1 and the timing numbers in table 2. The number of exemplars is kept small in these experiments as all the contesting methods scale linearly with the number of exemplars. To reduce this linear scaling one could either use the algorithms in a distributed setting proposed in [20] or combine with lazy-evaluation using a priority queue for marginal gains. For methods involving stochasticity (e.g. random subset selection in stochastic greedy, the proposed LRGR method, and k-means initialization) we report an average objective value and run-time over 10 trials.

We can see that the proposed low-rank greedy approach with random projections provides a very close approximation to the exact greedy objective values at an orders of magnitude faster time. We also see that among all the approximate algorithms the proposed method typically provides the best objective value (excluding the exact greedy method), and always within the top two results. The time is also competitive with other algorithms. K-means followed by finding cluster centroids is quite a good contender on small-dimensional data-sets, but its approximation quality can be poor on high-dimensional sparse data-sets. Furthermore, K-means assumes a Euclidean distance between data-points and does not apply to more general distance matrices.[7]. For KNN-greedy the time is dominated by the construction of the sparse nearest neighbors graph. Once the graph is constructed the algorithm is fast, and provides solutions which are quite reasonable although not as good as the proposed approach. The construction of the KNN graph can be accelerated using approximate

[5] Data is at http://www.math.uwaterloo.ca/tsp/world/.

[6] We use the low-rank matrix completion solver provided by the authors of [27].

[7] For example, k-means can not be used with spherical geodesic, transportation or string-edit distances, while there are no such constraints for exemplar clustering.

Table 1: Objective values achieved by greedy submodular algorithms with inner product similarity metric and 10 exemplars. Here d_{avg} is the average number of features per sample. Number of exemplars is 10. The best solution (after exact plain greedy which we aim to approximate) is marked with one star, and second-best with two stars. The proposed approach (LRGR) is typically most accurate, and is within the top 2 in all of our experiments.

Data	N (M)	d	$\mathbf{d_{avg}}$	RND	KmeansCtr	StochGR	KnnGR	LRGR	Greedy
Satimage	4435	36	36	3587.77	**3997.6***	3969.38	3977.13	**3983.4****	3976.42
Sector	6412	55197	163	381.5	667.01	717.96	**777.23****	**779.24***	787.53
Pendigits	7494	16	16	6925.22	**7171.33***	7122.27	7024.70	**7146.24****	7147.87
Pendigit-RF	7494	12891	100	2194.15	2962.44	2901.8	**2994.0****	**3004.27***	3015.23
RCV1	20242	47236	74	1217.67	1946.44	2003.26	**2175.85****	**2193.92***	2239.40
CodRNA	59535	8	8	53256.81	**56442.23***	55845.54	55411.42	**56133.72****	56146.50
CodRNA-RF	59535	7611	50	22290.93	**26353.32****	25909.36	n/a (mem)	**26533.53***	26746.18
Covtype	581012	54	11.9	490120.3	521045.25	**521188****	n/a (mem)	**523629.12***	n/a
Covtype-RF	581012	54509	50	86973.72	**136830.64****	136357.15	n/a (mem)	**146731.86***	n/a
World-Scale	1904711	20	20	6974224.6	n/a	**7309125.1****	n/a (mem)	**7408043.0***	n/a

Table 2: Timing results of greedy submodular algorithms with inner product similarity metric. We allowed up-to 1 hour for all the competing approaches.

Data	N (M)	d	$\mathbf{d_{avg}}$	KmeansCtr	StochGR	KnnGR	KnnGR-Ttl	LRGR	Greedy
Satimage	4435	36	36	0.085s	0.114s	0.178	1.36	0.15s	2.056s
Sector	6412	55197	163	2.613s	1.227s	0.366	7.49	4.82s	68.682s
Pendigits	7494	16	16	0.115s	0.232s	0.356	1.73	0.229s	5.430s
Pendigit-RF	7494	12891	100	2.240s	0.909s	0.313	6.46	3.24s	59.515s
RCV1	20242	47236	74	5.561s	2.106s	1.304	40.83	7.52s	375.067s
CodRNA	59535	8	8	0.637s	1.550s	3.974	19.77	1.82s	593.721s
CodRNA-RF	59535	7611	50	15.981s	5.818s	n/a	n/a	15.17s	2995.639s
Covtype	581012	54	11.94	9.107s	17.302s	n/a	n/a	24.17s	n/a
Covtype-RF	581012	54509	50	102.257s	37.214s	n/a	n/a	140.96s	n/a
World-Scale	1904711	20	20	n/a	53.4s	n/a	n/a	67.5s	n/a

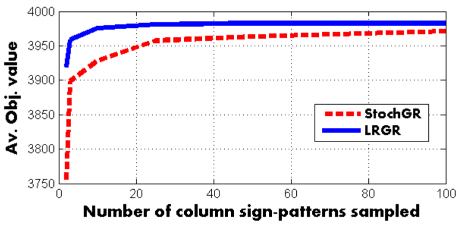

Figure 2: Av. objective values for LRGR and StochGR vs. num. sampled columns on Satimage data. 25 trials.

nearest neighbors (ANN) techniques, for example based on locality-sensitive-hashing (LSH) [9, 22], but that would also affect the quality of the results. Finally, it is quite clear that selecting random subsets is a rather poor (albeit very fast) data-summarization approach and any other method can provide a significant improvement.

The proposed low-rank greedy method achieved a slightly better objective than the exact greedy method that it tries to approximate in one experiment. This is not a contradiction as selecting a suboptimal exemplar in the current step may possibly improve the objective in further steps.

Dependence on the number of sampled columns. In Figure 2 we plot the average objective value achieved via Stochastic greedy and the proposed LRGR method vs. the number of sampled columns. The values are averages over 25 trials. We see that LRGR achieves better values by making better use of the samples, especially for smaller numbers of sampled columns.

6 Concluding Remarks

We developed a new approach for large-scale exemplar clustering based on submodular greedy maximization with structured similarity matrices. We propose a randomized sign-pattern sampling approach to approximate the bottleneck computation of finding the element with the maximal marginal gain. We establish accuracy guarantees, and evaluate the approach on large scale exemplar clustering problems on machine learning data-sets including up-to millions of data-points. Our current approach is serial and uses a single processor, but it can also be used as a sub-routine in previously proposed distributed approaches to extend exemplar clustering to problems of much larger scale.

We thank Baruch Schieber and Robert Hildebrand for helpful discussions.

References

[1] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proc. 18th ACM-SIAM symposium on Discrete algorithms*, 2007.

[2] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proc. of 20th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 671–680. ACM, 2014.

[3] F. Barahona and F. Chudak. Near-optimal solutions to large-scale facility location problems. *Discrete Optimization*, 2(1):35–50, 2005.

[4] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful? In *Database theory - ICDT99*. Springer, 1999.

[5] C. Blake and C. J. Merz. UCI repository of machine learning databases. 1998.

[6] E. Elhamifar, G. Sapiro, and R. Vidal. Finding exemplars from pairwise dissimilarities via simultaneous sparse recovery. In *Advances in Neural Information Processing Systems*, 2012.

[7] E. Elhamifar, G. Sapiro, A. Yang, and S. S. Sasrty. A convex optimization framework for active learning. In *Computer Vision (ICCV), 2013 IEEE Int. Conf. on*, pages 209–216. IEEE, 2013.

[8] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[9] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[10] C. Ho and S. Zimmerman. On the number of regions in an m-dimensional space cut by n hyperplanes. *Australian Math. Society Gazette*, 33(4), Sep. 2006.

[11] L. Kaufman and P. Rousseeuw. *Clustering by means of medoids*. North-Holland, 1987.

[12] A. Krause and D. Golovin. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, 3:19, 2012.

[13] N. Lazic, B. J. Frey, and P. Aarabi. Solving the uncapacitated facility location problem using message passing algorithms. In *Int. Conf. on Artificial Intelligence and Statistics*, pages 429–436, 2010.

[14] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proc. of the 49th Annual Meeting of the Assoc. for Computational Linguistics*, 2011.

[15] E. M. Lindgren, S. Wu, and A. G. Dimakis. Sparse and greedy: Sparsifying submodular facility location problems. In *Neural Information Processing Systems Workshop*, 2015.

[16] H. B. McMahan, G. Holt, D. Sculley, M. Young, Dietmar Ebner, Julian Grady, L. Nie, T. Phillips, E. Davydov, and D. Golovin. Ad click prediction: a view from the trenches. In *Proc. 19th ACM SIGKDD Int. Conf. on Knowledge discovery and Data mining*, 2013.

[17] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013.

[18] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, pages 234–243. Springer, 1978.

[19] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrak, and A. Krause. Lazier than lazy greedy. *arXiv preprint arXiv:1409.7938*, 2014.

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

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

[22] B. Neyshabur and N. Srebro. On symmetric and asymmetric LSH for inner product search. In *International Conference on Machine learning*, 2015.

[23] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2007.

[24] S. Si, C.J Hsieh, and I. Dhillon. Memory efficient kernel approximation. In *Proc. of the 31st Int. Conf. on Machine Learning*, 2014.

[25] V. Sindhwani, T. Sainath, and S. Kumar. Structured transforms for small-footprint deep learning. In *Advances in Neural Information Processing Systems*, pages 3070–3078, 2015.

[26] K. Wei, R. Iyer, and J. Bilmes. Fast multi-stage submodular maximization. In *Proc. of Int. Conf. on Machine Learning (ICML)*, 2014.

[27] H.F. Yu, Hsieh C.J., S. Si, and I.S. Dhillon. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *IEEE International Conference of Data Mining*, 2012.