Coresets for Estimating Means and Mean Square Error with Limited Greedy Samples

Saeed Vahidian
Electrical and Computer Engineering
University of California San Diego
San Diego, CA 92093

Baharan Mirzasoleiman
Computer Science
Stanford University
Stanford, CA 94305

Alexander Cloninger
Mathematics, HDSI
University of California San Diego
San Diego, CA 92093

Abstract

In a number of situations, collecting a function value for every data point may be prohibitively expensive, and random sampling ignores any structure in the underlying data. We introduce a scalable optimization algorithm with no correction steps (in contrast to FrankWolfe and its variants), a variant of gradient ascent for coreset selection in graphs, that greedily selects a weighted subset of vertices that are deemed most important to sample. Our algorithm estimates the mean of the function by taking a weighted sum only at these vertices, and we provably bound the estimation error in terms of the location and weights of the selected vertices in the graph. In addition, we consider the case where nodes have different selection costs and provide bounds on the quality of the low-cost selected coresets. We demonstrate the benefits of our algorithm on the semi-supervised node classification of graph convolutional neural network, point clouds and structured graphs, as well as sensor placement where the cost of placing sensors depends on the location of the placement. We also elucidate that the empirical convergence of our proposed method is faster than random selection and various clustering methods while still respecting sensor placement cost. The paper concludes with validation of the developed algorithm on both synthetic and real datasets, demonstrating that it outperforms the current state of the art.

1 INTRODUCTION

In many problems in sociology, finance, computer science, and operations research, we have networks of interconnected entities and pairwise relations between them. A problem that arises often in practice is calculating the expected value of a variable in the form of the sum of the values of a smooth function on the nodes of a graph. For example, in semi-supervised learning with Graph Neural Networks (GCNs), the generalization error is specified as the average of the loss functions associated with all the nodes in a graph. Before the elections, opinion polls are usually designed to represent the opinions of a networked population about the candidates in expectation [28]. In social networks, having a small average distance to other individuals in the network is considered a key factor to be influential [1]. In many environmental monitoring, knowing average temperature, humidity and water quality of various regions allow for taking preventive actions against forest fires and water contamination [20, 36]. Finally, in health care, monitoring various health measures such as a population's average blood pressure, weight, cholesterol level, allow for designing health strategies and disease prevention actions [27].

In real-world networks containing millions of nodes and billions of edges, it is impractical to evaluate the function at every single node. Therefore, an important question is how to select a small representative subset (coreset) of nodes from a million-node graph such that the weighted sum of function values sampled at the nodes of the subset can be a good estimate of the sum of function values over the entire graph [5]. Another constraint that often arises in practice is that evaluating the function may incur a cost that is not necessarily equal for all the nodes in the graph. For example, placing sensors in certain areas may be more expensive than others [17]. Similarly, measuring blood pressure of people in hard-to-reach areas is more expensive. Hence, we wish to find a small representative weighted subset of nodes subject to a limited budget.

There are main challenges in finding such a small coreset. First, the selected subset and the corresponding weights should provide a bound on the estimation error of the first moment (mean) of the function at all the nodes in the graph. Moreover, the method should be simple to imple-

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

ment, computationally inexpensive, and have theoretical guarantees relating coreset size to both computational complexity and the quality of approximation. Finally, different nodes may have non-uniform cost, and hence we need to be able to bound the error of estimating the mean while finding a low-cost solution.

Very recently, the authors in [27] considered this problem and provided an upper-bound on the estimation error of the mean of the function evaluated over the entire graph with a weighted subset of functions at nodes of an arbitrary subset. The provided quadrature-type bound can be used to bound any mean estimation problem in which the function is sufficiently low-frequency, i.e., the function can be expressed in terms of a small number of eigenfunctions (with large eigenvalue) of a lazy random walk transition matrix on the underlying graph. This includes problems such as measuring average blood pressure in a database [27] and subsampled kernel two sample testing [9]. Intuitively, the placements that minimize the quadrature-type bound have the property that the random walks starting from every node in the subset and weighted by its corresponding weight, overlap very little. However, the problem of finding the near-optimal subset and the associated weights by minimizing the upper-bound has remained unaddressed.

In this work, we address the question of finding a coreset of nodes that minimizes the estimation error of the expected function value over the entire graph, by minimizing the upper-bound provided in [27]. Inspired by the recent work of [6] on Bayesian coreset constructions, we propose a greedy algorithm to find a small subset of nodes and their weights that closely approximate the first moment of the function over the entire graph. Moreover, we consider an extended problem of having each sampled node come with a non-uniform cost, a problem that arises in applications such as sensor placement, marketing, and other knapsack type problems. We extend our algorithm to this setting, and characterize through a simple parameter the error in estimating the mean of the function one is willing to tolerate in order to seek a low cost solution.

The paper is organized as follows. In Section 3, we describe the mathematical framework of our problem. In Section 4, we frame the greedy optimization algorithm for selecting points and weights, both for equal cost of placement and when there is a placement cost associated. In Section 5, we prove bounds on the convergence of our algorithm and bound the mean error of a smooth function in terms of the algorithmically selected points. In Section 6, we demonstrate the success of our algorithm over random sampling and several benchmark unsupervised learning, semi-supervised learning, and clustering algorithms for a number of different applications.

2 RELATED WORK

The ever increasing size of modern datasets motivated data reduction techniques as a preprocessing step to speed up subsequent optimization problems. Existing graph summarization methods mainly focus on obtaining sparse subgraphs that can be used to approximate properties of the original graph (degree distribution, size distribution of connected components, diameter, or community structure). Core techniques include graph clustering or community detection methods [14, 23], bit compression-based methods [33], sparsification-based [35] and sketching methods [3, 26], and influence-based methods [31]. While these methods maintain structural properties of the original graph, they cannot guarantee that an algorithm working on the summary provides a solution close to the solution found based on the entire data. For instance, graph summarization algorithms cannot guarantee that the function sampled at the selected points has similar statistics to the function evaluated on the entire network.

In contrast, coresets are weighted subsets of the data, which guarantee that for the specific problem at hand, models fitting the coreset also provide a good fit for the original dataset. This approach has been successfully applied to a variety of problems including K-means and K-median clustering [16], mixture models [30], low rank approximation [10], spectral approximation [2, 25], Nystrom methods [2, 32], and Bayesian inference [6]. Coreset construction methods traditionally perform importance sampling with respect to sensitivity score, defined as the importance of the point with respect to the objective function we wish to minimize, to provide high-probability solutions [16, 30, 10]. Greedy algorithms, which are special cases of the Frank-Wolfe algorithm, were described more recently to provide worst-case guarantees for problems such as support vector machines [8], and Bayesian inference [6]. In this work, we propose a greedy algorithm to construct coresets for estimating the first moment of a function defined on the nodes of a large graphs.

3 GENERAL FRAMEWORK

Here, we aim to choose a subset of vertices and weights to be the best representative of the graph. More specifically, we assume that the dataset V have some geometric structure encoded in a graph $G = (V, E)$, where V and E denote the set of nodes and edges. The problem is how to choose a subset $S \subseteq V$ of vertices and weights $w_s > 0$ for every $s \in S$ in order to approximate the mean μ_f with a weighted approximation $\widehat{\mu}_f$, where

$$\mu_f = \frac{1}{|V|} \sum_{v \in V} f(v), \qquad \widehat{\mu}_f = \sum_{s \in S} w_s f(s). \tag{1}$$

The graph can either be given a priori, or constructed on a point cloud $V \subset \mathbb{R}^d$ via a kernel $K : V \times V \to \mathbb{R}_+$. We must assume that the function $f : V \to \mathbb{R}$ must have some relationship to the graph, or else the optimal sampling would be a random search. In our context, this assumption takes the form that the function can be expressed in terms of a small number of eigenfunctions (with large eigenvalue) of a lazy walk graph transition matrix on the graph G.

Definition 3.1. *The lazy random walk graph* [1] *transition matrix P on $G = (V, E)$ is constructed by*

$$P = \frac{1}{d_{max}}(A - D) + I,$$

where A denote the (weighted) symmetric adjacency matrix of G such that $A_{ij} = A_{ji} \geq 0$ is the weight of the edge connecting nodes i and j, D is a diagonal matrix with $D_{ii} = \sum_j A_{ij}$, $d_{max} = \max_i D_{ii}$, and I is the identity matrix.

Definition 3.2. *A function $f : V \to \mathbb{R}$ is in P_λ for a lazy walk transition matrix P, with eigendecomposition $P = U\Lambda U^*$, if*

$$f = \sum_{|\lambda_i| > \lambda} b_i U_i,$$

where λ_i and U_i are the eigenvalues and eigenvectors of P, $b_i > 0$ is a weight, and $0 \leq \lambda \leq 1$ is a parameter controlling the degree of smoothness. P_λ is the subspace spanned by the eigenfunctions of P associated with eigenvalue $\geq \lambda$.

Spectral smoothness on graphs, as in Definition 3.2, is necessitated by the fact that there is no traditional notion of gradients on graphs. Theres a large body of work that shows that eigenfunctions of the kernel with large eigenvalue are of lower frequency than eigenvectors with small eigenvalue [11, 15]. Thus spectrally band-limited functions must themselves be smooth [27]. Our ability to approximate the mean of the function decays as the frequency of the function increases, since the function becomes more chaotic. This is a fundamental limitation, as functions unrelated to the underlying graph structure cannot be approximated with a coreset in any way better than random sampling.

We can also consider the additional constraint that choosing every nodes $v \in V$ may come with a cost $C : V \to \mathbb{R}_+$. We seek an algorithm that will choose the subset of nodes $S \subseteq V$ in a way that is

- Greedy in order to quickly choose and add additional points,
- Minimizes the error in estimation of the mean of f, and
- Incorporates the cost $C(v)$ by choosing low-cost points to the subset.

The motivation for this framework and use of the lazy walk graph transition matrix comes out of the work in [27], in which it was noted that the mean error in f can be bounded in terms of the choice of points and weights, as in the following

$$\forall f \in P_\lambda, \left| \frac{1}{n} \sum_{v \in V} f(v) - \sum_{s \in S} w_s f(s) \right| \leq \|f\|_{P_\lambda} \min_{\ell \in N} \frac{1}{\lambda^\ell} \left(\left\| P^\ell \sum_{s \in S} w_s \delta_s \right\|_2^2 - \frac{1}{n} \right)^{\frac{1}{2}}, \tag{2}$$

for $\ell \in \mathbb{N}$, $0 < \lambda < 1$, and w_s summing to 1. Moreover, n is the number of vertices of the graph, and δ_s is the binary vector taking value zero at every index except for s. This result implies that minimizing the right hand side in terms of w_s and S will yield a stronger bound on the moment estimation of f. $P^\ell \delta_s$ is the probability distribution of a random walker starting in s after ℓ jumps. Therefore, intuitively the subset S that minimize the quadrature-type bound have the property that the random walks starting from every node s in the subset and weighted by its corresponding weight w_s overlap very little.

3.1 PROBLEM DEFINITION

Similar to the concept of duality in convex optimization, in order to achieve best results for the upper bound in (2) we shall minimize it. Therefore, the optimization problem can be formulated in the following from

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \|f\|_{P_\lambda} \frac{1}{\lambda^\ell} \left(\left\| P^\ell \sum_{s \in S} w_s \delta_s \right\|_2^2 - \frac{1}{n} \right)^{\frac{1}{2}} \\ \text{subject to} \quad & |S| \leq K, \; S \subset V, \\ & \sum_{s \in S} w_s = 1, \quad w_s > 0. \end{aligned} \tag{3}$$

where K is the maximum number of selected vertices. Several discrete and continuous methods are relevant when solving the preceding problem in (3). However, the main issue with such an optimization is the constraint of choosing $S \subset V$, which leads to a difficult combinatorial optimization problem. For example, one approach related to (3) is that of computing a cardinality constrained minimization by solving sparse principal component analysis

[1] A lazy random walk is a walk on a graph with self loops. Here the probability of taking self loop is inversely proportional to the degree of the node.

(PCA) problem [12]. However, solving the sparse PCA optimization problem entails semidefinite relaxation and a greedy algorithm to calculate a full set of good solutions, which is very expensive with total complexity of $O(n^3)$ [13].

A better way to view this problem is in terms of an L_2-minimization problem on P. Because $\|f\|_{P_\lambda}$ and λ are properties of the function f we're analyzing (see Def. 3.2) and independent of the choice of points and weights, we will drop these terms in discussion of the optimization scheme, when appropriate. Similarly, we will drop the dependence on ℓ for notational simplicity and simply deal with an arbitrary lazy random walk matrix P. This is not an issue as P^ℓ is also a lazy random walk transition matrix, with eigenvalues λ^ℓ.

We can also rewrite our cost as a matrix multiplication $Pw = P\sum_s w_s\delta_s$, and define our target function to be the normalized ones vector $\frac{1}{n}\mathbb{1}$. We note that the cost function of (3) can be reframed by the above notational changes, as well as bringing the $\frac{1}{n}$ inside the norm, to arrive at

$$\begin{aligned}\left\|P\sum_{s\in S} w_s\delta_s\right\|_2^2 - \frac{1}{n} &= \|Pw\|_2^2 + \frac{1}{n} - 2\left\langle Pw, \frac{1}{n}\mathbb{1}\right\rangle \\ &= \left\|Pw - \frac{1}{n}\mathbb{1}\right\|_2^2,\end{aligned}$$

where the first equality comes from the fact that $\langle Pw, \mathbb{1}\rangle = 1$, and the second equality comes from the fact that $\|\frac{1}{n}\mathbb{1}\|_2^2 = \frac{1}{n}$ and completing the square. To this end, (3) can be posed in an equivalent form as

$$\begin{aligned}\underset{w}{\text{minimize}} \quad & \left\|Pw - \frac{1}{n}\mathbb{1}\right\|_2 \\ \text{subject to} \quad & \sum_i \mathbb{1}[w_i > 0] \le K \\ & w_i \ge 0\end{aligned} \tag{4}$$

Problem (4) shifts the original mean bound in (3) into a constrained least squares problem for finding the optimal weights w. Due to the shifting of $\frac{1}{n}\mathbb{1}$ inside the norm, we are no longer constrained to have $\sum_{i\in V} w_i = 1$ (discussed in detail in Appendix C). With that being said, our algorithm still has a weight normalization β^* that arises in (6) and will push $\|w\|_1$ close to 1.

The optimization problem (4) we construct has been considered previously [34, 21], however these have limitations in the context of our graph problem. Briefly, [34] considers cardinality regularized loss function minimization subject to simplex constraints, which yields a computational complexity $O(n^4)$ for our graph problem. Similarly, [21] has theoretical guarantees only under the restricted isometry property [7]. Moreover, it assumes the size of the subset K is known, which may vary in an a posteriori fashion in our applications. Finally, we extend the literature by providing guarantees on the convergence rate explicitly in terms of the number of selected elements and the rate at which the error decreases as we sample more coreset points.

We also note that the Problem (4) can be solved by relaxing the nonconvex cardinality constraint $\sum_i \mathbb{1}[w_i > 0] \le K$ to a simplex constraint $\sum_i \|P_i\| w_i = \sum_i \|P_i\|$ for the columns P_i of the matrix P, and using the FrankWolfe (FW) algorithm that iteratively chooses the point most aligned with the residual error. However, there are some problems for which FW performs very poorly for any number of iterations because FW must scale the objective function in Problem (4) suboptimally by $\sum_i w_i\|P_i\|$ rather than $\|\frac{1}{n}\mathbb{1}\|$, in order to maintain feasibility [6]. In this paper, we mainly focus on the above-mentioned constrained optimization problem. In the following section, we provide a new radial optimization algorithm for problem (4) and demonstrate that it yields theoretical guarantees at a significantly reduced computational cost. More importantly, in contrast to FW and its extensions [22], the algorithm developed in this work has no correction steps and geometric error convergence.

4 OPTIMIZATION ALGORITHM

The optimization problem in (4), despite involving a least squares cost function, is nonconvex in w due to the cardinality constraint. Inspired from [6], without any loss of generality, the weights, w could be scaled by an arbitrary constant $\beta \ge 0$ without affecting feasibility. This motivates rewriting (4) as

$$\begin{aligned}\underset{w,\beta}{\text{minimize}} \quad & \left\|\beta Pw - \frac{1}{n}\mathbb{1}\right\|_2 \\ \text{subject to} \quad & \sum_i \mathbb{1}[w_i > 0] \le K \\ & w_i \ge 0, \beta \ge 0\end{aligned} \tag{5}$$

Following [6] we now begin by solving the optimization problem in β. We define β^* as the solution to the problem (5) given w which can be computed analytically as

$$\begin{aligned}\beta^* &= \frac{\|\frac{1}{n}\mathbb{1}\|}{\|Pw\|}\max\left\{0, \left(\frac{Pw}{\|Pw\|}\right)^T\left(\frac{\frac{1}{n}\mathbb{1}}{\|\frac{1}{n}\mathbb{1}\|}\right)\right\} \\ &= \frac{1}{n\|Pw\|}\max\left\{0, \left(\frac{Pw}{\|Pw\|}\right)^T\mathbb{1}\right\}.\end{aligned} \tag{6}$$

Substituting β^* in the objective above and expanding the square, the weighted subset of nodes (coreset) can be

found by solving:

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \frac{1}{n}\left(1-\max\left\{0,\left(\frac{Pw}{\|Pw\|}\right)^T\mathbb{1}\right\}\right) \\ \text{subject to} \quad & \sum_i \mathbb{1}[w_i>0] \le K \\ & w_i \ge 0 \end{aligned} \tag{7}$$

This result shows that the minimum of the problem in (7) occurs by alignment of the vectors Pw and $\frac{1}{n}\mathbb{1}$ independent of their norm. Finally, we define $P(w) = \sum_i w_i \frac{P_i}{\|P_i\|}$, and $P^* = \frac{\frac{1}{n}\mathbb{1}}{\left\|\frac{1}{n}\mathbb{1}\right\|} = \frac{1}{\sqrt{n}}\mathbb{1}$. With that in mind, we can reformulate (7) in an equivalent maximizing problem as in the following

$$\begin{aligned} \underset{w}{\text{maximize}} \quad & P(w)^T P^* \\ \text{subject to} \quad & \sum_i \mathbb{1}[w_i>0] \le K \\ & \|P(w)\| = 1 \\ & w_i \ge 0 \end{aligned} \tag{8}$$

According to the constraints in (8), we are optimizing the objective over a unit hypersphere rather than the simplex. Before solving the problem in (8), we extend it to a more general case where nodes have different selections costs, and then we provide a new algorithm for solving it.

4.1 OPTIMIZATION WITH SELECTION COST

In many applications, selecting some reference points representing the whole data involve some factors such as cost of selection associated to each data. For example, placing sensors in certain regions may be more expensive than others. Similarly, measuring blood pressure of people in hard-to-reach areas is more expensive. In problems such as these, it is natural to seek a trade-off between the two goals of minimizing the error (selecting nodes $S \subseteq V$ that maximize alignment of $P(w)$ and P^*) and the cost of choosing nodes in S. To this end, we incorporate a new parameter C into the problem controlling the cost associated with each node. In what follows, we will focus on the reparameterized maximization problem, which can be written as:

$$\begin{aligned} \underset{w}{\text{maximize}} \quad & P(w)^T P^* - \lambda C(S) \\ \text{subject to} \quad & |S| \le K \text{ for } S = \{i : w_i > 0\} \\ & \|P(w)\| = 1 \\ & w_i \ge 0 \end{aligned} \tag{9}$$

Now we provide a greedy algorithm, sample cost greedy iterative geodesic ascent (SCGIGA), for the above optimization problem. At every iteration, the algorithm finds the point indexed by v^* for which the geodesic between $P(w)$ and $P(v^*)$ is most aligned with the geodesic between $P(w)$ and P^*. We then find the set of all vertices for which the alignment is within κ-percent of the alignment of v^*, for a parameter $0 < \kappa \le 1$. Among such points, we add the vertex with minimum cost to the solution. Once the point has been added, the algorithm reweights the coreset and iterates. SCGIGA detailed in Algorithm 1 outlines how to solve the optimization problem in (9). It is noteworthy that SCGIGA is a general algorithm which is also valid for the case where there are equal costs associated with every data point. This corresponds to $\kappa = 1$ in our settings.

A benefit of the greedy approach is that increasing the number of coreset points to $K+1$ simply requires adding the next point for geodesic ascent and marginally changing the weights. This is unlike [21], in which changing to the $K+1$ simplex requires recomputing the optimization scheme and in no way guarantees keeping the previous K selected data points. Similarly, this formulation allows us to easily incorporate nonuniform cost of sampling points.

We note that the most expensive computation in Algorithm 1 is $\langle P_v, P(w)\rangle$ across v, but that $P(w)$ is the sum of at most K vectors and each loop only adds one additional vector. Thus we can store previous inner products and only take different weighted combinations on each iteration, so each loop only requires n inner products. Hence, Algorithm 1 has average complexity $O(Knm)$, where m is the average sparsity of a column.

We will establish the convergence guarantees and rate in Section 5, and connect this convergence to bounding the estimate of the mean of f as in (2). But first, we wish to establish that incorporating cost of sampling into the optimization does not significantly impact the resulting minimum value, and thus results in a close to optimal greedy bound on the error in estimating the mean of f. The gap can be characterized by the one parameter κ to be chosen by the user, and choosing the minimal cost point among the set of vertices similar to v^* only scales the resulting bound by a factor of κ.

Theorem 1. *Let C^k_{max} be the sum of the k largest elements of C. If we choose $\lambda \le \frac{1-\kappa}{C^k_{max}\min\|P_i\|\sqrt{n}}$, then the solution to* (9)*, $P(w^*)$, satisfies*

$$P(w^*)^T P^* \ge \kappa \max_w P(w)^T P^*.$$

The proof can be found in the Appendix.

Theorem (1) implies we will never incur too much loss to the original objective by incorporating cost of selecting the nodes that minimize the error. Similarly, this implies that we can make every greedy choice and step in whatever fashion is deemed best for cost, as long as the choice

Algorithm 1 Algorithm of SCGIGA

Initialization $w_0 \leftarrow 0$
$\forall v \in V, P_v \leftarrow \frac{P_v}{\|P_v\|}$
for $k \in \{0, ..., K\}$ **do**
  $a_k \leftarrow \frac{\mathbb{1} - \langle \mathbb{1}, P(w_k) \rangle P(w_k)}{\|\mathbb{1} - \langle \mathbb{1}, P(w_k) \rangle P(w_k)\|}$
  $\forall v \in V, a_{kv} \leftarrow \frac{P_v - \langle P_v, P(w_k) \rangle P(w_k)}{\|P_v - \langle P_v, P(w_k) \rangle P(w_k)\|}$
  $v^* \leftarrow \arg\max_{v \in V} {a_k}^T a_{kv}$
    ▷ find vertex that maximizes alignment
  $W \leftarrow \left\{ v \in V | {a_k}^T a_{kv} \geq \kappa {a_k}^T a_{kv^*} \right\}$
    ▷ find vertices within κ-percent of max
  $v_k \leftarrow \arg\min_{v \in W} C_v$
    ▷ find vertex in W with min cost
  $\zeta_0 = \langle \frac{1}{\sqrt{n}} \mathbb{1}, P_{v_k} \rangle$
  $\zeta_1 = \langle \frac{1}{\sqrt{n}} \mathbb{1}, P(w_k) \rangle$
  $\zeta_2 = \langle P_{v_k}, P(w_k) \rangle$
  $\delta_k \leftarrow \frac{\zeta_0 - \zeta_1 \zeta_2}{(\zeta_0 - \zeta_1 \zeta_2) + (\zeta_1 - \zeta_0 \zeta_2)}$
    ▷ choose the step size
  $w_{k+1} \leftarrow \frac{(1-\delta_k) w_k + \delta_k 1_{v_k}}{\left\| (1-\delta_k) P(w_k) + \delta_k P_{v_k} \right\|}$
    ▷ update the weight
  end
end for
$w = \beta w_k$ ▷ Scale weights by β
$S = \{ v \in V | w_v > 0 \}$
Sample f at nodes in S
$\widehat{\mu}_f = \sum_{v \in S} w_v f(v)$
return $\widehat{\mu}_f, w$

is within κ of the optimal step direction.

5 THEORETICAL ASPECTS

Here we examine the guarantees that Algorithm 1 yields for bounding the error in estimating the mean of $f \in P_\lambda$, where P_λ is the subspace spanned by the eigenfunctions of P associated with eigenvalues $\geq \lambda$. We will derive the general theorem for arbitrary κ, and as a special case we recover the results when all the nodes have equal selection cost ($\kappa = 1$).

Theorem 2. *Let $f \in P_\lambda$ with mean μ_f as in* (1)*, and assume there is a cost for selecting every node $v \in V$, i.e., $C(v) : V \to \mathbb{R}_+$ and a slack parameter κ. If we choose the set of points S and weights w_s using Algorithm 1 such that $|S| = K$, then*

$$\left| \mu_f - \sum_{s \in S} w_s f(s) \right| \leq \frac{\|f\|_{P_\lambda}}{\lambda^\ell} \frac{\eta \, v_K}{\sqrt{n}},$$

where $v_K = O((1 - \kappa^2 \epsilon^2)^{K/2})$ for some ϵ and $\eta = \sqrt{1 - \kappa^2 \max_{i \in V} \left\langle \frac{P_i}{\|P_i\|}, \frac{1}{\sqrt{n}} \mathbb{1} \right\rangle^2}$.

The proof can be found in the Appendix.

The main ideas of this theorem are three-fold. The first series of lemmas that must be proved are extensions of the core lemmas in [6], which establish that the error is a contractive map after each update, which is later used in a fixed point theorem to show convergence and rate. The extensions we make here are: 1) generalize their results to graphical geometries, rather than the log-likelihood construction proposed in [6], and 2) allow for a κ relaxation of the greedy choice and prove how this relaxation affects the contractive mapping.

The second main idea behind this theorem is to show that the κ-relaxation does not significantly affect the fixed point argument to show convergence of the cost-aware greedy algorithm as the number of chosen points grows. This again borrows from [6] in this more general setting, but the argument effectively comes down to propagating the additional error accrued by the cost-aware algorithm.

The final main idea behind this theorem is connecting the coreset choice to the first moment quadrature bound in [27]. This comes from a shifting of the bound in (2) to allow for weights to not necessarily have to sum to 1, and the recognition that this bound for general points and weights chosen is equivalent to the bound being minimized in (4).

We wish to note that the bound established in Theorem 2 may not be sharp for the first few points greedily sampled, but does become sharp asymptotically due to fixed point convergence. The bound on the first moment estimate of f in terms of (2) is close to sharp if there is a spectral gap at λ [27].

As a particular case of this theorem, when we always choose the optimal greedy node independent of cost, we recover the following guarantee.

Corollary 3. *Let $f \in P_\lambda$, and choose the set of points S and weights w_s using Algorithm 1 such that $|S| = K$. Then*

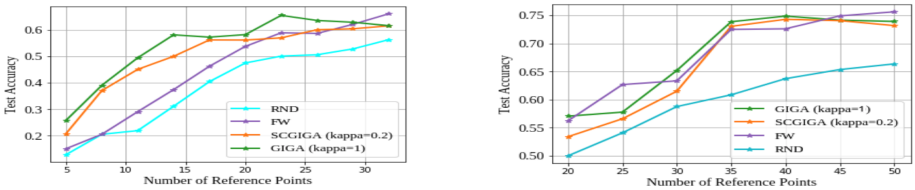
$$\left| \mu_f - \sum_{s \in S} w_s f(s) \right| \leq \frac{\|f\|_{P_\lambda}}{\lambda^\ell} \frac{\eta \, v_K}{\sqrt{n}},$$

where $v_K = O((1 - \epsilon^2)^{K/2})$ for some ϵ and $\eta = \sqrt{1 - \max_{i \in V} \left\langle \frac{P_i}{\|P_i\|}, \frac{1}{\sqrt{n}} \mathbb{1} \right\rangle^2}$.

The proof of this corollary is a special application of Theorem 2, which is proved in the Appendix.

6 EMPIRICAL EVIDENCE

In this section, we evaluate our model on several sets of experiments. In order to compare both cases of our algo-



Stochastic block model Cora citation dataset

Figure 1: Classification accuracy obtained from a GCN model after a number of semi-supervised training iterations for different algorithms.

rithm, i.e., when there are non-uniform costs on selecting different data points ($\kappa \neq 1$) and when there are equal costs associated with every data point ($\kappa = 1$) we consider a fixed cost on each data randomly generated from a uniform distribution over $[0, 1)$. Similarly, in all examples involving point clouds, the generated graph comes from a K-nearest neighbor construction with 10 nearest neighbors.

6.1 GRAPH CNN CLASSIFICATION

One example of the importance of bounding means comes in semi-supervised learning. In such a problem, the goal is to approximate a function $h : V \to \mathbb{R}^m$ using a parametric model h_θ (such as the graph convolutional network). The goal is to minimize

$$L_{\theta,V} = \frac{1}{|V|} \sum_{v \in V} \|h_\theta(v) - h(v)\|^2,$$

where L is a loss function, chosen judiciously depending upon the application. However, in the event that sampling h is expensive, the goal is to choose a coreset $S \subset V$ in order to estimate $L_{\theta,V}$ with a quadrature formula and the empirical risk

$$L_{\theta,S} = \sum_{v \in S} w_v \|h_\theta(v) - h(v)\|^2.$$

The quadrature error in this context, $|L_{\theta,S} - L_{\theta,V}|$, is exactly the *generalization error*, and is now re-expressed as the estimation of the mean of the function $f(v) = \|h_\theta(v) - h(v)\|^2$. Theorem 2 applies in the situation that $h \in P_\lambda$ and that h_θ is sufficiently regularized to satisfy $h_\theta \in P_\lambda$. We note that the $\|\cdot\|^2$ remains mostly low-frequency as the frequency of pointwise product of eigenfunctions can be bounded [29]. This implies that sampling h only at S, and training h_θ on S, one can bound the generalization error on $V \setminus S$ using Theorem 2.

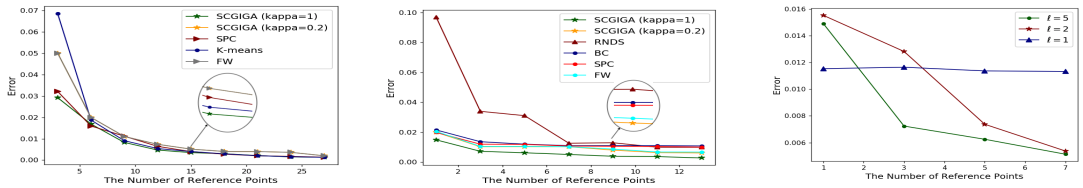
Given the above discussion, we seek to test our algorithm on semi-supervised document classification in citation networks datasets, namely, Cora and semi-supervised node classification in a stochastic block model graph with 200 vertices and 10 clusters. We train a two-layer graph convolutional neural network (GCN) as described in [19] and we follow the same pre-processing techniques as well. We compare against the same baseline methods as in the experiments presented in our paper tested on Cora dataset which is a real citation network dataset with $2,708$ nodes and $5,429$ edges as well as a stochastic cluster-based graph datasets.

A GCN takes a feature matrix and an adjacency matrix as inputs and for every vertex of the graph produces a vector, whose elements correspond to the score of belonging to different classes. An identity matrix is added to the original adjacency matrix in order to enforce self loops.

The neural network is trained in a semi-supervised setting, where the network is fed with the feature and adjacency matrices of the entire graph while the loss is only computed on the labeled vertices. Here the labeled vertices are the subset of vertices that are picked by the proposed method on the normalized adjacency matrix. We train both datasets for a maximum of 100 epochs using Adam [18] with a learning rate of 0.01 and early stopping with a window size of 10, i.e. we stop training if the validation loss does not decrease for 10 consecutive epochs (as was done in [19]). The results are summarized in Figure 1. Despite small fluctuations due to random initialization, as expected, the test accuracy tends to increase as more labeled points are utilized for training. Further, as can be seen from the figure our proposed algorithm, SCGIGA, has superior performance in selecting the subset of data that comprises the most representative points of clusters. Lastly, because of the existence of outliers in a random graph, the accuracy of the proposed algorithm starts to improve slowly at about 60% accuracy. However, we note that the model is trained with only 10% of data which was considered to be the labeled ones, so this also implicitly suggests that our algorithm successfully picks out the most informative nodes.

6.2 MEAN FUNCTION ON CLUSTERED DATA

A standard unsupervised learning task is to learn clusters from data, either on a graph or a point cloud, and use those clusters to select points and weights for averaging a function. Standard clustering algorithms include K-



Gaussian model | Stochastic Block Model | Stochastic Block Model for ℓ

Figure 2: The error comparison of estimating the mean of the function defined on the clusters (the two left ones), and the impact of the shaping parameter ℓ on the performance (Right one).

means clustering and spectral clustering [2].

The first experiment we run is on Gaussian model with three components. The components have the mean vectors of $[1\ -3]$, $[-3\ \ 2]$, $[3\ \ 0]$, and all have the covariance matrix of the identity matrix, I. The components contain 20%, 30%, and 50% of the data. The function we choose to model is a simple smooth function that is an indicator function of the small cluster (1 on the small cluster, 0 on other clusters). The results in Fig. 2 for the Gaussian Model show the error comparison of three algorithms including FW, K-means and spectral clustering (SPC) versus the number of clusters (centroids) or reference points. The performance of the algorithms is quantified by an error metric which is defined as the $\text{Err} = \left|\sum_{|S|=k} a_s f\left(S\right) - \mu_f\right|^2$. For fairness to comparable algortihms, a_s in K-means, FW and SPC is defined as the ratio of the data in each cluster to the whole data, while $a_s = w_s$ in our algorithm. Here, f is the the indicator function on the small cluster and μ_f is the mean of the function.

As is evident from the figure, our algorithm outperforms K-means, FW and spectral clustering for the whole range of the number of reference points. As can bee seen in this figure the maximum number of the reference data is 14 out of 10000 and our results reveal that the cost of the optimal solution (C_{COS}) is 5.920, while the cost we got with our cost aware algorithm, i.e., the cost of sub-optimal solution (C_{CSO}) is 0.106. The more surprising observation in these figures is that even in the case where a fixed cost associated with each data in which results in our algorithm to yield a sub-optimal solution (the solution which is in the κ percent of the optimal) our algorithm continues to do very well and work better than standard algorithms. Also note that for only 3 reference points, which would lead to an ideal coreset of one point per cluster, SCGIGA considerably reduces the error compared to the competing algorithms.

[2] In multivariate statistics spectral clustering techniques gets rid of some of the eigenvalues of the similarity matrix of the data to perform nonlinear dimensionality reduction before clustering in fewer dimensions.

We consider another experiment on clustered graphical data. In Fig. 2 a stochastic block model with three clusters is designed where the first cluster contained 10% of the data and the other two clusters contain 50% and 40% of the data. Then we define an indicator function on the small cluster, and we look for the average function value on these three clusters. As can be seen from these figures, our algorithm estimates the mean very well while random sampling (RNDS) needs more reference points to catch up the mean. More importantly, in this experiment by selecting 28 reference data, $C_{\text{COS}} = 15.159$ while $C_{\text{CSO}} = 0.075$.

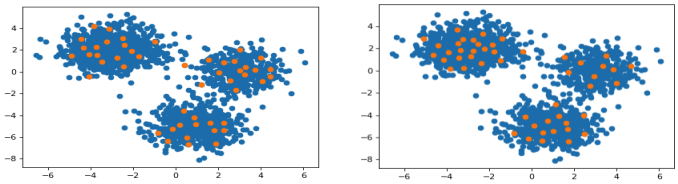
Further, Fig. 2 sketches the impact of the shaping parameter, ℓ from (3), in our formulation on the error behavior of the same stochastic block model. Clearly, incorporating a multi-step kernel yields much better performance in estimating the cluster coresets and weights.

Finally Fig. 3 shows three Gaussian clusters with the same mean vectors and covariance matrices as in the first experiment is considered. These two figures demonstrate the way that the proposed algorithm selects points when there is varying cost associated with each point and the special case of equal costs for choosing each point. In the figure, when there is a higher average cost of sampling on the largest cluster compared to the smallest, the algorithm adapts to resample in a way not simply proportional to the number of points in each cluster. In contrast, when no cost is associated the number of selected points are obviously less than that of the largest component.

6.3 SHORTEST PATH ON GRAPH

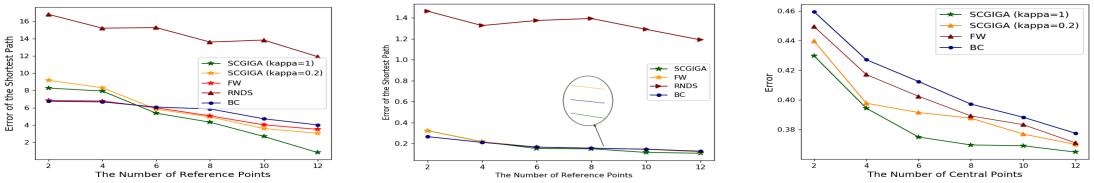
In this set of experiments, we assume that a graph $\mathcal{G}$ or an adjacency matrix is given. The goal is to find the average distance from a vertex to the rest of the graph, where the distance between two points is computed using Dijkstras algorithm [3]. In this experiment, rather than computing the shortest path between a given node and every other node on the graph, we calculate the shortest path between

[3] Dijkstras algorithm is a greedy nature algorithm that looks for the minimum weighted vertex on every iteration.



Cost associated No cost associated

Figure 3: 2% of coresets selected two cases, variable sampling cost with $\kappa = 0.2$ (left) and uniform cost (right). The upper left-hand component, the middle component, and the lower component contain respectively 50%, 20%, 30% of the data, and the average cost per data in each component is 3.25, 0.51, and 2.04, respectively.



A Powerlaw tree graph A random generated graph Facebook ego network

Figure 4: Shortest path comparison of different algorithms on different graphs.

the given node and just the reference points S. This is of particular cost benefit on trees that have large diameter (maximum distance between two nodes) but low average path distance (e.g., trees, powerlaw graphs, etc).

Fig. 4 shows the results of the comparison of our algorithm, FW, Betweenness Centrality (BC) and random sampling (RS) on a Powerlaw Tree graph [4]. The error is defined as the difference between the weighted average distance of each vertex of the graph from the reference vertices and the average distance of each vertex from all the vertices of the graph. Fig. 4 also demonstrates the same results for a randomly generated graph. In these two figures, both cases i.e., when fixed but different costs are associated with each data (vertex) and when equal costs on the data are included.

Ego Networks In this experiment we consider a special type of network called an Ego Network. In an Ego Network, there is a central vertex (ego vertex) which the network highly depends on or revolves around. We consider the real-world Facebook Ego Networks dataset with 4,964 nodes [24]. The dataset contains the aggregated network of some users' Facebook friends. In this dataset, vertices represent individuals on Facebook, and edges between two users mean they are Facebook friends. The Ego Network connects a Facebook user to all of his Facebook friends and are then aggregated by identifying individuals who appear in multiple Ego Network. Algorithms such as betweenness centrality (BC) were proposed to measure the importance (centrality) of a user in the network. These algorithms select a user as a central one by looking at how many shortest paths pass through that user (vertex). The more shortest paths that pass through the user, the more central the user is in the Facebook network. We run our algorithm on the Facebook dataset to choose the central nodes. We then compare our algorithm with the BC and FW in terms of the error performance metrics described in the first experiment. The results in Fig. 4 show that the developed algorithm in this work perform better in selecting the central points in the Facebook graph.

7 DISCUSSION AND CONCLUSIONS

In this paper, we introduced a scalable, and theoretically-sound algorithm for minimizing the estimation error of the expected value of a function over entire graph. Our proposed method can be applied to semi-supervised classification in graph neural networks (GCNs), social network graphs, point clouds, and on many other applications where a similarity metric between data points can be defined. We provided theoretical guarantees on the convergence of the estimated mean and validated its efficiency empirically on real and synthetic datasets. Our work also opens up the notion of optimizing when there is a non-uniform cost of sampling, and provides a simple parameter for the trade off between accuracy and cost.

ACKNOWLEDGEMENTS

AC was funded by NSF-DMS grants 1819222 and 2012266, and Sage Foundation Grant 2196. BM was partially supported by SNSF P2EZP2_172187.

References

[1] Y. Abbasi, P. L. Bartlett, V. Kanade, Y. Seldin, and C. Szepesvari. Online learning in markov decision processes with adversarially chosen transition probability distributions. In *NeurIPS*. 2013.

[2] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *Journal of the ACM*, 2004.

[3] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *ACM SIGMOD PoDS*. ACM, 2012.

[4] W. Aiello, F. C. Graham, and L. Lu. A random graph model for power law graphs. *Experimental Mathematics*, 10(1):53–66, 2001.

[5] A. Anis, A. Gadde, and A. Ortega. Efficient sampling set selection for bandlimited graph signals using graph spectral proxies. *IEEE Trans. Sig. Proc.*, 2016.

[6] T. Campbell and T. Broderick. Bayesian coreset construction via greedy iterative geodesic ascent. In *ICML*, 2018.

[7] E. J. Candes et al. The restricted isometry property and its implications for compressed sensing. *Comptes rendus mathematique*, 2008.

[8] K. L. Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM TALG*, 6(4):63, 2010.

[9] A. Cloninger. Bounding the error from reference set kernel maximum mean discrepancy. In *arXiv preprint arXiv:1812.04594*, 2018.

[10] M. B. Cohen, C. Musco, and C. Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *ACM-SIAM SODA*, pages 1758–1777. SIAM, 2017.

[11] R. R. Coifman and S. Lafon. Diffusion maps. *ACHA*, 21(1):5–30, 2006.

[12] A. d'Aspremont, F. R. Bach, and L. E. Ghaoui. Optimal solutions for sparse principal component analysis. *JMLR*, 2008.

[13] A. d'Aspremont, L. E. Ghaoui, M. I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 2007.

[14] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[15] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *ACHA*, 30(2):129–150, 2011.

[16] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *ACM Symposium on Theory of Computing*, pages 291–300. ACM, 2004.

[17] R. K. Iyer and J. A. Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *NeurIPS*. 2013.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[19] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[20] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 2008.

[21] A. Kyrillidis, S. Becker, V. Cevher, and C. Koch. Sparse projections onto the simplex. In *ICML*, pages 235–243, 2013.

[22] S. Lacoste-Julien and M. Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *NeurIPS*, pages 496–504, 2015.

[23] A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Physical review E*, 80(5):056117, 2009.

[24] J. Leskovec and J. J. Mcauley. Learning to discover social circles in ego networks. In *NeurIPS*, pages 539–547. 2012.

[25] M. Li, G. L. Miller, and R. Peng. Iterative row sampling. In *IEEE FoCS*, pages 127–136. IEEE, 2013.

[26] E. Liberty. Simple and deterministic matrix sketching. In *ACM SIGKDD Inter. Conf. on KDD*, pages 581–588. ACM, 2013.

[27] G. Linderman and S. Steinerberger. Numerical integration on graphs: Where to sample and how to weigh. 2020.

[28] W. Lippmann. *Public opinion*. Routledge, 2017.

[29] J. Lu, C. D. Sogge, and S. Steinerberger. Approximating pointwise products of laplacian eigenfunctions. *Journal of Functional Analysis*, 277(9):3271–3282, 2019.

[30] M. Lucic, M. Faulkner, A. Krause, and D. Feldman. Training gaussian mixture models at scale via coresets. *JMLR*, 18(1):5885–5909, 2017.

[31] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsification of influence networks. In *ACM SIGKDD Inter. Conf. on KDD*, 2011.

[32] C. Musco and C. Musco. Recursive sampling for the nystrom method. In *NeurIPS*, pages 3833–3845, 2017.

[33] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *ACM SIGMOD Inter. Conf. on Management of data*. ACM, 2008.

[34] M. Pilanci, L. E. Ghaoui, and V. Chandrasekaran. Recovery of sparse probability measures via convex programming. In *NeurIPS*, pages 2420–2428, 2012.

[35] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[36] L. Yu, N. Wang, and X. Meng. Real-time forest fire detection with wireless sensor networks. In *International Conference on Wireless Comm., Networking and Mobile Computing*, volume 2, pages 1214–1217. IEEE, 2005.