Data Sketching for Faster Training of Machine Learning Models

Baharan Mirzasoleiman  
Department of Computer Science  
Stanford University  
baharann@cs.stanford.edu

Jeff Bilmes  
Department of Electrical Engineering  
University of Washington, Seattle  
bilmes@uw.edu

Jure Leskovec  
Department of Computer Science  
Stanford University  
jure@cs.stanford.edu

Abstract

Many machine learning problems reduce to the problem of minimizing an expected risk, defined as the sum of a large number of, often convex, component functions. Iterative gradient methods are popular techniques for the above problems. However, they are in general slow to converge, in particular for large data sets. In this work, we develop analysis for selecting a subset (or sketch) of training data points with their corresponding learning rates in order to provide faster convergence to a close neighborhood of the optimal solution. We show that subsets that minimize the upper-bound on the estimation error of the full gradient, maximize a submodular facility location function. As a result, by greedily maximizing the facility location function we obtain subsets that yield faster convergence to a close neighborhood of the optimum solution. We demonstrate the real-world effectiveness of our algorithm, SIG, confirming our analysis, through an extensive set of experiments on several applications, including logistic regression and training neural networks. We also include a method that provides a deliberate deterministic ordering of the data subset that is quite effective in practice. We observe that our method, while achieving practically the same loss, speeds up gradient methods by up to 10x for convex and 3x for non-convex (deep) functions.

1 Introduction

Training machine learning models often reduces to the problem of minimizing the sum of a large number of convex component functions. The standard gradient method can find the minimizer of this problem, but requires repeated computations of the full gradient—sum of the gradients of all the components functions—and is therefore prohibitive for massive data sets. Incremental gradient (IG) methods are a popular alternative, in particular in the context of large-scale learning. IG methods take sequential steps along the gradient of every component function in a cyclic order [1, 2]. The analysis of IG is valid regardless of the order of processing the component. Hence, at every cycle, the components can be processed based on a predefined deterministic order, or they can be chosen with replacement uniformly at random (SGD).

While every SGD update provides an unbiased estimate of the full gradient, the randomness introduces variance, and therefore stochastic gradient methods are in general slow to converge. On the other hand, performance of IG has shown to be pretty sensitive to the order of processing the functions, and poor deterministic orders might incur exponentially worse convergence rates than

Preprint. Under review.
randomized methods [1,3]. While finding a favorable ordering for IG has been a long standing open question, variance reduction and accelerated gradient methods to speed up SGD have been subject to a large body of recent studies [4,5,6,7,8].

In this paper, we develop a principled method, data Sketching for Incremental Gradient descent (SIG), for finding an ordered representative subset $S \subseteq V$, and the corresponding sequence of per-element stepsizes to speed up IG method on massive data sets. Our contributions are two fold. We first show that subsets that minimize the upper-bound on the error of estimating the full gradient maximize a submodular facility location function. As a result, by maximizing the facility location function, we obtain weighted subsets that guarantees fast convergence to a close neighborhood of the optimum solution. While the size of the subsets that can closely approximate the full gradient depends on the structural properties of the data, for a subset of size $|S| = r$, our method speeds up the convergence of IG by $|V|/|S| = n/r$ times. Then, we propose a \textit{deliberate} data-driven deterministic order (or permutation) $(\sigma_1, \cdots, \sigma_r)$ over the subset $S$ where $\sigma_i \in S$ for $i \in [r]$, that results in further speeding up IG in practice. Indeed, the components that reduce the estimation error the most are the medoids (exemplars) of all the components in the gradient space. Hence, our method orders the elements of the subset in a way that the weighted sum of the first $i$ elements $\{\sigma_1, \cdots, \sigma_i\}$ for every $i \in [r]$ best estimates the full gradient, and the rest of the elements $\{\sigma_{i+1}, \cdots, \sigma_r\}$ further fine-tune the estimation.

Theoretically, we prove that for a $\mu$-strongly convex risk function and a subset $S$ that estimates the full gradient by an error of at most $\epsilon$, IG with a diminishing stepsize of $\alpha_k = \alpha/k^s$ for $0 < s < 1$ and $0 < \alpha$, converges to an $R/\mu$ neighborhood of the optimum solution. Here, $R = \min\{D, (r\gamma_{\max}C + \epsilon)/\mu\}$ for a dataset of diameter $D$, an upper bound $C$ on the norm of the gradients, and largest per-element stepsizes $\gamma_{\max}$. Moreover, we show that for a $\mu$-strongly convex risk and smooth component functions, IG with the same diminishing stepsize rule on subset $S$ converges to a $\epsilon/\mu$ neighborhood of the optimum solution. While we do not formally prove the improvement of the convergence rate resulted by our proposed ordering, we practically show that our proposed ordering is extremely effective in reducing the variance of IG iterates and improve its convergence rate.

We demonstrate the effectiveness of our SIG algorithm, confirming our theoretical analysis, via an extensive set of experiments on several applications, including logistic regression (a convex optimization problem) as well as training neural networks (non-convex optimization problems). We observe that our SIG method, while achieving practically the same loss as the standard stochastic gradient descent, speeds up gradient methods by up to 10x for convex and 3x for non-convex loss functions. We also demonstrate that the deliberate ordering scheme of the SIG algorithm significantly improves convergence time.

### 2 Related Work

Due to the simplicity and popularity of IG methods, they are supported by convergence analyses under various conditions [9, 10, 11, 12, 13] (see [14] for a survey). Most closely related to our paper are the analysis of [11] which provides a $O(1/\sqrt{R})$ convergence rate under a strong convexity assumption, and the analysis of [12] that proves a $O(1/k^s)$ convergence rate for $s \in (0, 1]$ under an additional smoothness assumption for the component function gradients.

Order of processing the components has been numerically observed to significantly affect the convergence rate of IG. However, finding a favorable ordering of the components for IG has been a long standing open question. Among the few related results are that of [15] showing that without-replacement sampling improves convergence of IG for least means squares problem, and the very recent result of [3] showing that Random Reshuffling (RR) method has a better convergence rate compared to its with-replacement counterpart, stochastic gradient descent (SGD).

Techniques for speeding up SGD, are mostly focused on variance reduction techniques [4,5,6,7,8], and accelerated gradient methods when the regularization parameter is small [16,17,18]. Of particular interest are the results of [7,8] showing that neighborhood structure can be exploited to further reduce the variance of stochastic gradient descent and improve its running time.

To the best of our knowledge, this study provides the first analysis of subset selection and ordering of the components to speed up IG methods.
3 Incremental Gradient Methods and Risk Minimization

We consider the following additive risk minimization problem

$$x_* \in \arg\min_{x} f(x), \quad f(x) := \sum_{i \in V} f_i(x) + r(x), \quad f_i(x) = l(x, (a_i, y_i)),$$  \tag{1}

where the objective $f$ is the sum of a large number of convex component functions $f_i : \mathbb{R}^d \to \mathbb{R}$, and a $\mu$-strongly convex regularizer $r$. $V = \{1, 2, \ldots, n\}$ is an index set of the training data, and every function $f_i$ is associated with a training example $(a_i, y_i)$, where $a_i \in \mathbb{R}^d$ is the feature vector of data point $i \in V$, and $y_i$ is its label.

Incremental gradient (IG) methods are core algorithms for solving Problem (1) and are widely used and studied. IG aims at approximating the standard gradient method by sequentially stepping along the gradient of the component functions $f_i$ in a cyclic order. Starting from an initial point $x_0 \in \mathbb{R}^d$, it makes $k$ passes over all the $n$ components. At every cycle $k \geq 1$, it iteratively updates $x^k_i$ based on the gradient of $f_i$ for $i = 1, \ldots, n$ by an stepsize $\alpha_k > 0$. Formally,

$$x^k_i = x^{k-1}_i - \alpha_k \nabla f_i(x^{k-1}_i), \quad i = 1, 2, \ldots, n,$$  \tag{2}

with the convention that $x^{k+1}_0 = x^k_n$.

When the component functions are not smooth, we can replace gradients with a subgradient and the corresponding method is called the incremental subgradient method. Using the update relation (2), for each $k \geq 1$, we can write down the relation between the outer cycle as

$$x^{k+1}_0 = x^k_0 - \alpha_k \sum_{i=1}^n \nabla f_i(x^{k-1}_i),$$  \tag{3}

where $\sum_{i=1}^n \nabla f_i(x^{k-1}_i)$ is the aggregated component gradients and serves as an approximation to the full gradient $\nabla f(x^k_0) = \sum_{i=1}^n \nabla f_i(x^k_0)$.

The convergence rate of IG is known to be quite sensitive to the order of processing the elements [19]. If problem-specific knowledge can be used to find a favorable order $\sigma$ (defined as a permutation of $\{1, 2, \ldots, n\}$), IG can be updated to process the functions according to this order, i.e.,

$$x^k_i = x^{k-1}_i - \alpha_k \nabla f_{\sigma_i}(x^{k-1}_i), \quad i = 1, 2, \ldots, n$$  \tag{4}

However, in general a favorable order is not known in advance, and a common approach is sampling the function indices with replacement from the set $\{1, 2, \ldots, n\}$ and is called the Stochastic Gradient Descent (SGD) method, a.k.a. the Robbins-Monro algorithm [20] (also see [21][13][22][23]).

4 Data Sketching for Incremental Gradient Descent (SIG)

As discussed in the previous section, incremental gradient methods aim at estimating the full gradient $\sum_{i=1}^n \nabla f_i(x^k_0)$ by iteratively stepping along the gradient of every component $\sum_{i=1}^n \nabla f_i(x^{k-1}_i)$ within each cycle $k$. Therefore, if we can find a small subset such that the weighted sum of the gradient of its components closely approximates the full gradient, then when we apply IG to the elements of the subset, we should converge to an approximately optimal solution. If the subset is small but still representative (so the number of cycles is roughly the same), convergence to the optimum will be much faster.

Specifically, our goal in SIG is to find the smallest subset $S \subseteq V$ and corresponding per-element stepsizes (weights) $\gamma > 0$ that approximate the full gradient with an error at most $\epsilon > 0$ for all the possible values of $x \in \mathcal{X}$. That is, we wish to solve the following minimization problem:

$$S^* = \arg\min_{S \subseteq V, \gamma_j \geq 0 \forall j} |S|, \text{ s.t. } \| \sum_{i=1}^n \nabla f_i(x) - \sum_{j \in S} \gamma_j \nabla f_j(x) \| \leq \epsilon, \forall x \in \mathcal{X}. \tag{5}$$

Given and using such an $S^*$ and associated weights $\{\gamma\}_i$, our updates will be similar to the full gradient regardless of $x$. 

3
At the same time, we may wish to find a favorable ordering $\sigma^* \in \Gamma_S$ (where $\Gamma_S$ is the set of all permutations of $\{1, 2, \cdots, |S|\}$) on the elements of $S$ that provides fast convergence for IG methods. Every updates in IG is conditioned on all the past steps. As a result, if we order the components such that the sum of the gradients of the currently processed components provides the closest approximation to (i.e., is most representative of) the full gradient (meaning for each $q \in [r]$, we have $\| \sum_{j=1}^q \gamma_{\sigma_j} \nabla f_{\sigma_j}(x) - \sum_{i=1}^q \nabla f_i(x) \| \leq \| \sum_{j=1}^q \gamma_{\sigma_j} \nabla f_{\sigma_j}(x) - \sum_{i=1}^q \nabla f_i(x) \|$ for any other ordering $\sigma$), fast convergence is expected. Then, the IG iterates using the ordering $\sigma^*$ of the subset $S$ become:

$$x_i^k = x_{i-1}^k - \alpha_k \gamma_{\sigma} \nabla f_{\sigma}(x_{i-1}^k), \quad i = 1, 2, \cdots, |S|$$

(6)

In the following, we first discuss how we upper-bound the estimation error in Eq. (5). Then, we introduce our SIG algorithm to find the smallest ordered subset and the corresponding per-element stepsizes that guarantees (as we show below) fast convergence for IG.

### 4.1 Upper-bound on the Estimation Error

Let $S = \{s_1, \cdots, s_r\} \subseteq V$ be a subset of $r$ arbitrary component functions. Furthermore, assume that there is a mapping $\varsigma : V \rightarrow S$ that assigns every function $f_i$ to one of the elements in $S$, i.e., $\varsigma(i) = s_j \in S$. Let $C_j = \{i \in [n] | \varsigma(i) = s_j\}$ be the set of all the components $f_i$ that are assigned to $s_j \in S$, and $n_j = |C_j|$ be the number of such components. Then, we can write

$$\sum_{i=1}^n \nabla f_i(x) = \sum_{i=1}^n (\nabla f_i(x) - \nabla f_{\varsigma(i)}(x) + \nabla f_{\varsigma(i)}(x))$$

(7)

$$= \sum_{i=1}^n (\nabla f_i(x) - \nabla f_{\varsigma(i)}(x) + \sum_{j=1}^r n_j \nabla f_{s_j}(x)).$$

(8)

Subtracting and then taking the norm of the both sides, we get an upper bound on the error of representing the full gradient with the weighted sum of the gradients of the functions in $S$, i.e.,

$$\| \sum_{i=1}^n \nabla f_i(x) - \sum_{j=1}^r n_j \nabla f_{s_j}(x) \| \leq \sum_{i=1}^n \| \nabla f_i(x) - \nabla f_{\varsigma(i)}(x) \|,$$

(9)

where the right had side results from the triangle inequality. The upperbound in Eq. (9) is minimized when $\varsigma$ assigns every $i \in V$ to an element in $S$ with largest gradient similarity, or minimum euclidean distance between the gradient vectors. I.e., $\varsigma(i) = \arg\min_{s_j \in S} \| \nabla f_i(x) - \nabla f_{s_j}(x) \|$. Hence,

$$\min_{S \subseteq V} \left\{ \sum_{i=1}^n \nabla f_i(x) - \sum_{j=1}^r n_j \nabla f_{s_j}(x) \right\} \leq \min_{s_j \in S} \left\{ \sum_{i=1}^n \| \nabla f_i(x) - \nabla f_{s_j}(x) \| \right\}.$$

(10)

The right hand side of Eq. (10) is minimized when the sum of euclidean distances between the gradients of the functions assigned to $s_j \in S$ and $\nabla f_{s_j}(x)$ is minimized. In other words, $S$ is the set of $r$ medoids (exemplars) for all the components in the gradient space.

The size of the smallest weighted subset $S$ that can closely approximate the full gradient depends on the structural properties of the data. By solving the following minimization problem, we obtain the smallest weighted subset $S^*$ that approximates the full gradient by an error of at most $\epsilon$:

$$S^* = \arg\min_{S \subseteq V} |S|, \quad \text{such that} \quad L(S) = \sum_{i=1}^n \min_{s_j \in S} \| \nabla f_i(x) - \nabla f_{s_j}(x) \| \leq \epsilon \quad \forall x \in \mathcal{X}.$$

(11)

In the following section, we discuss efficient algorithms to find approximate solutions for the above minimization problem.

We note that the upper bound in Eq. (10) is tight as shown by the following example. Consider two sets of functions $f_1(x) = \cdots = f_{n/2}(x) = Rx/2$, and $f_{n/2+1}(x) = \cdots = f_n(x) = -Rx/2$. The optimum set $S = \{s_1\}$ of cardinality 1 that minimizes the upper bound in Eq. (10) is $R$ away from the actual mean and hence $\sum_{i=1}^n \| \nabla f_i(x) - \nabla f_{s_1}(x) \| = nR$. On the other hand, for $|S| = 2$, both sides of the inequality are $0$. 

4
At every step, the greedy algorithm selects an element that reduces the upper bound on the estimation error the most. Adding the element with largest marginal gain $\Delta_F(s|S_{i-1})$ improves our estimation from the mean by an amount bounded by the marginal gain. In fact, as long as the marginal gains of facility location are considerably large, we need more elements to improve our estimation of the full gradient. Having found $S$, the weight $\gamma_j$ of every element $j \in S$ is the number of components that are closest to it in the gradient space. The pseudocode is outlined in Algorithm 1.

Algorithm 1 SIG (Data Sketching for Incremental Gradient Descent)

Input: Set of component functions $f_i$ for $i \in V = \{1, \ldots, n\}$.

Output: Subset $S \subseteq V$ with corresponding per-element step sizes $\gamma_i$, and an ordering $\sigma$.

1: $S \leftarrow \{\}$, $s_0 = 0$.
2: while $F(S) < L(\{s_0\}) - \epsilon$ do
3:     $S_i = S_{i-1} \cup \{\arg\max_{s \in V} \Delta_F(s|S_{i-1})\}$
4: end while
5: for $i = 1$ to $|S|$ do
6:     $\gamma_j = \sum_{i=1}^{n} 1[j = \arg\min_{s_j \in S} \| \nabla f_i(x) - \nabla f_{s_j}(x) \| ]$
7:     $\sigma_i = s_i$
8: end for

Note that Eq. (11) requires that the error is bounded for all $x \in \mathcal{X}$. For several classes of convex problems, including linear regression, ridge regression, logistic regression, and regularized support vector machines (SVMs), we have that $\| \nabla f_i(x) - \nabla f_j(x) \| \leq O(\|x\|) \| a_i - a_j \|$ [8, 7]. Assuming $\|x\|$ is bounded for all $x \in \mathcal{X}$, upper-bounds on the euclidean distances between the gradients can be precomputed.

4.2 The SIG Algorithm

Previously we showed that by solving the optimization problem (11), we obtain a weighted subset of elements that closely approximates the full gradient. Here, we discuss how we can efficiently solve problem (11) to get a weighted subset with a favorable ordering on its elements to speed up IG.

Although the optimization problem (11) is NP-hard, it can be turned into a submodular cover problem, for which efficient approximation algorithms exist. Formally, $F$ is submodular if $F(S \cup \{s\}) - f(S) \geq F(T \cup \{s\}) - F(T)$, for any $S \subseteq T \subseteq V$ and $s \in V \setminus T$. We denote the marginal utility of an element $s$ w.r.t. a subset $S$ as $\Delta(s|S) = F(S \cup \{s\}) - F(S)$. Function $F$ is called monotone if $\Delta(s|S) \geq 0$ for any $s \in V \setminus S$ and $S \subseteq V$. Submodular cover problem is defined as finding the smallest set $S$ that achieves a utility $Q$. More precisely,

$$S^* = \arg\min_{S \subseteq V} |S|, \text{ such that } F(S) \geq Q.$$  \hspace{1cm} (12)

Although finding $S^*$ is NP-hard, for many classes of submodular functions [24, 25], a simple greedy algorithm is known to be very effective. Having a ground set $V$ of elements, the greedy algorithm starts with the empty set $S_0 = \emptyset$, and at each iteration $i$, it chooses an element $s \in V$ that maximizes $\Delta(s|S_{i-1})$, i.e., $S_i = S_{i-1} \cup \{\arg\max_{s \in V} \Delta_F(s|S_{i-1})\}$. Greedy gives us a logarithmic approximation, i.e. $|S| \leq (1 + \log |V|)\min_{S} |S^*|$. The computational complexity of the greedy algorithm for finding a subset of size $r$ is $O(|V| \cdot r)$. However, its running time can be significantly improved using lazy evaluation and distributed implementations [26, 27, 28].

By introducing an auxiliary element $s_0$ (e.g., $s_0 = 0$) we can turn $L(S) = \sum_{i=1}^{n} 1[\min_{s_j \in S} \| \nabla f_i(x) - \nabla f_{s_j}(x) \| ]$ in Eq. (11) into a monotone submodular facility location function,

$$F(S) = L(\{s_0\}) - L(S \cup \{s_0\}),$$  \hspace{1cm} (13)

where $L(\{s_0\})$ is a constant. In words, $F$ measures the decrease in the estimation error associated with the set $S$ versus the estimation error associated with just the auxiliary element. It is easy to see that for suitable choice of $s_0$, maximizing $F$ is equivalent to minimizing $L$. Therefore, we apply the greedy algorithm to approximately solve the following problem to get the subset $S$ defined in (11):

$$S^* = \arg\min_{S \subseteq V} |S|, \text{ such that } F(S) \geq L(\{s_0\}) - \epsilon.$$  \hspace{1cm} (14)

At every step, the greedy algorithm selects an element that reduces the upper bound on the estimation error the most. Adding the element with largest marginal gain $\Delta_F(s|S_{i-1})$ improves our estimation from the mean by an amount bounded by the marginal gain. In fact, as long as the marginal gains of facility location are considerably large, we need more elements to improve our estimation of the full gradient. Having found $S$, the weight $\gamma_j$ of every element $j \in S$ is the number of components that are closest to it in the gradient space. The pseudocode is outlined in Algorithm 1.
5 Convergence Rate Analysis of SIG

We analyze the convergence rate of IG applied to the subset $S$ found by SIG with the corresponding per-element step sizes. If the iterates $\{x_0^n, x_1^n, \ldots, x_k^n\}$ are uniformly bounded, i.e. there exists a non-empty compact Euclidean ball $X \subset \mathbb{R}^d$ that contains all the iterates, the optimal solution set $x_*$ is nonempty. Under the above assumption, we show that on a subset $S$ returned by SIG that approximates the full gradient by an error of at most $\epsilon$, IG converges to a close neighborhood of the optimal solution.

5.1 Convergence Rate for Strongly Convex Functions

We first provide the convergence analysis for the case where the expected risk is strongly convex, i.e. $\forall x, y \in \mathbb{R}^d$ we have that $f(y) - f(x) \geq (\nabla f(x), y - x) + 1/2 \|y - x\|^2$.

**Theorem 1.** Assume that $\sum_{i=1}^n f_i(x)$ is strongly convex and $S$ is a subset of size $r$ such that $\| \sum_{i \in V} \nabla f_i(x) - \sum_{j \in S} \gamma_j \nabla f_j(x) \| \leq \epsilon, \forall x \in X$. Then for the iterates $\{x_k = x_0^n\}$ generated by applying IG to $S$ with per-cycle stepsize $\alpha_k = \alpha/k^s$ with $\alpha > 0$ and $s \in [0, 1]$, we have for $k \to \infty$

1. if $s = 1$, then $\|x_k - x_*\| \leq 2\epsilon R/\mu + r^2 \gamma_{\text{max}}^2 C^2/(k\mu)$.
2. if $0 < s < 1$, then $\|x_k - x_*\| \leq \epsilon R/\mu$.
3. if $s = 0$, then $(1 - \alpha_k\mu)^{k-1}\|x_0 - x_*\|^2 + 2\epsilon R/\mu + \alpha_k r^2 \gamma_{\text{max}}^2 C^2/\mu$,

where $C$ is an upper bound on the norm of the gradients, i.e. $\max_{i \in V} \sup_{x \in X} \|\nabla f_i(x)\| \leq C$, $\gamma_{\text{max}} = \max_{j \in S} \gamma_j$ is the largest per-element step size, and $R = \min\{D, (r\gamma_{\text{max}} C + \epsilon)/\mu\}$ for a dataset with diameter $D$.

Hence, for $s > 0$, IG on the subset $S$ converges to an $\epsilon R/\mu$ neighborhood of the optimal solution, at a rate $O(1/\sqrt{k})$. However, since every cycle is $|V|/|S| = n/r$ times faster, we get an speedup of $n/r$.

All the proofs can be found in the Appendix.

5.2 Convergence Rate for Smooth and Strongly Convex Functions

If in addition to strong convexity of the expected risk, each component function has a Lipschitz gradient, i.e. $\forall x, y \in X, i \in [n]$ we have $\|\nabla f_i(x) - \nabla f_i(y)\| \leq \beta_i \|x - y\|$, then we get the following results about the iterates generated by applying IG to the weighted subset $S$ returned by SIG.

**Theorem 2.** Assume that $\sum_{i=1}^n f_i(x)$ is strongly convex and let $f_i(x)$, $i = 1, 2, \ldots, n$ be convex and twice continuously differentiable component functions with Lipschitz gradients on $X$. Assume there is a subset $S$ such that $\| \sum_{i \in V} \nabla f_i(x) - \sum_{j \in S} \gamma_j \nabla f_j(x) \| \leq \epsilon, \forall x \in X$. Then for the iterates $\{x_k^n\}$ generated by applying IG to $S$ with per-cycle stepsize $\alpha_k = \alpha/k^s$ with $\alpha > 0$ and $s \in [0, 1]$, we have for $k \to \infty$

1. if $s = 1$, then $\|x_k - x_*\| \leq (\epsilon + \alpha^2 \beta C r \gamma_{\text{max}}^2 / k)/\mu$.
2. if $0 < s < 1$, then $\|x_k - x_*\| \leq \epsilon / \mu$.
3. if $s = 0$, then $\|x_k - x_*\| \leq (1 - \alpha\mu)^k \|x_0 - x_*\| + \epsilon / \mu + \alpha \beta C r \gamma_{\text{max}}^2 / \mu$,

where $\beta = \sum_{i=1}^n \beta_i$ is the sum of gradient Lipschitz constants of the component functions.

The above theorem shows that for $s > 0$, IG applied to $S$ converges to an $\epsilon/\mu$ neighborhood of the optimum solution, with a rate of $O(1/k^s)$. However, every cycle is $n/r$ times faster than IG applied to the entire dataset. Therefore, we get an speedup of $n/r$. As shown in our experiments, in real datasets small weighted subsets constructed by SIG provide a close approximation of the full gradient. Hence, applying IG to the weighted subsets returned by SIG provides a solution of the same or higher quality compared to the solution obtained by applying IG to the whole dataset, in a considerably less amount of time.
6 Experiments

In our experimental evaluation we wish to address the following questions: (1) How do loss and accuracy of IG applied to the subsets returned by SIG compare to loss and accuracy of IG applied to the entire data; (2) How small is the size of the subsets that we can select with SIG and still get a comparable performance to IG applied to the entire data; (3) how does the ordering affect the performance of IG on the subset; And (4) how well does SIG scale to large datasets, and extend to non-convex problems. To this end, we apply SIG to several convex and non-convex problems.

We compare the performance of our SIG method (using constant and diminishing learning rates $\alpha_k$) to the performance of (a) IG on the entire dataset and (b) Adaptive Moment Estimation (Adam) on random subsets of the same size as those found by SIG. Adam computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients \cite{Adam}. Therefore, it provides a strong baseline for comparing the effectiveness of the per-element stepsizes computed by SIG. In our experiments, we report the run-time as the wall-clock time for subset selection with SIG, plus minimizing the loss using IG or Adam with the specified learning rates. Finally, for the classification problems, we separately select subsets from each class while maintaining the class ratios in the whole data, and apply IG to the union of the subsets.

6.1 Convex Experiments

In our convex experiments, we apply L2-regularized logistic regression: $f_i(x) = \ln(1 + \exp(-x^T a_i y_i)) + 0.5\lambda x^T x$ to classify the following three datasets from LIBSVM: (1) covtype.binary including 581,012 data points of 54 dimensions, (2) SensIT including 78,823 training and 19,705 test data points of dimension 100, and (3) Ijcnn1 including 49,990 training and 91,701 test data points of 22 dimensions. As covtype do not come with labeled test data, we randomly split the training data into halves to make the training/test split (training and test sets are consistent for different methods). For the convex experiments, $\lambda$ is set to $10^{-5}$.

Figures 1a and 1d compare training loss residual and test error rate of IG on the subsets of size 10% and 20% of covtype selected by SIG (with corresponding per-element stepsizes) to that of Adam and IG on the entire dataset. It can be seen that SIG outperforms the baselines by achieving a similar loss and error rate, but much faster. In particular, we obtain a 9.5x and 8x speedup from applying IG on the subsets of size 10% and 20% selected by SIG. Figures 1b and 1e show similar behavior on SensIT with three classes of ratios 1:1:2. We note that, SIG outperforms the baselines while having a lower variance. A similar behavior can be observed for Ijcnn1 with a high class-imbalanced ratios of 95:5, as shown in Figures 1c and 1f. Here, we got a speedup of 9x and 5.5x for subsets of size 10% and 20% respectively.
6.2 Non-convex Experiments

Our non-convex experiments involves applying SIG to train the following two neural networks: (1) Our smaller network is a fully-connected hidden layer of 100 nodes and ten softmax output nodes; sigmoid activation and L2 regularization with $\lambda = 0.0001$ and mini-batches of size 10 on MNIST dataset of handwritten digits containing 60,000 training and 10,000 test images. (2) Our large neural network is ResNet-56 for CIFAR10 with convolution, average pooling and dense layers with softmax outputs and L2 regularization with $\lambda = 0.0002$ CIFAR 10 includes 50,000 training and 10,000 test images from 10 classes, and we used mini-batches of size 128. Both MNIST and CIFAR10 are normalized into $[0, 1]$ by division with 255.

Unlike convex problems where we use SIG to find a subset before starting the optimization and do not change the subset across cycles of IG, for non-convex problems we apply SIG to select a subset at the beginning of every cycle and train only on the selected subset with the corresponding per-element stepsizes in every cycle. Here, we consider the gradient of the loss w.r.t. the last layer of the network, as it is shown that the variation of the gradient norms is mostly captured by these variables \cite{30}. Fig. 3a, 3b, 3c shows loss, and error rate for training a 1 layer neural net on MNIST. Fig. 3d, 3e, 3f show similar quantities for training ResNet-56 on CIFAR10. For both problems, we
used a constant learning rate of 1e-2 for applying IG on subsets found by SIG. On the other hand, we let Adam to tune its learning rate, starting with the same rate of 1e-2. Interestingly, SIG is every effective for subset selection for the non-convex problem of training neural networks by selecting subsets of size 30%, 40%, and 50% and achieves a speedup of 2x to 3x.

7 Conclusion

We developed analysis for selecting an ordered subset of data points with their corresponding per-element stepsizes to speed up iterative gradient (IG) method. In particular, we showed that weighted subsets that minimize the upper-bound on the estimation error of the full gradient, maximize a submodular facility location function. Hence, we can obtain an ordered subset of data points with their corresponding learning rates using a greedy algorithm. We showed that IG on subsets returned by our method, SIG, while achieving practically the same loss, runs up to 10x faster on convex and up to 3x on non-convex problems. We further demonstrated the effect of the ordering found by the greedy algorithm on the convergence rate of IG.

Acknowledgment. JL is a Chan Zuckerberg Biohub investigator. This research has been supported in part by SNSF P2EZP2_172187, NSF OAC-1835598, DARPA MCS, DARPA ASED, ARO MURI, Boeing, Docomo, Hitachi, Huawei, JD, Siemens and Stanford Data Science Initiative.

References

[1] Angelia Nedić and Dimitri Bertsekas. Convergence rate of incremental subgradient algorithms. In Stochastic optimization: algorithms and applications, pages 223–264. Springer, 2001.

[2] Mert Gürbüzbalaban, Asu Ozdaglar, and Pablo Parrilo. Why random reshuffling beats stochastic gradient descent. arXiv preprint arXiv:1510.08560, 2015.

[3] Mert Gurbuzbalaban, Asuman Ozdaglar, and Pablo A Parrilo. On the convergence rate of incremental aggregated gradient algorithms. SIAM Journal on Optimization, 27(2):1035–1048, 2017.

[4] Nicolas L Roux, Mark Schmidt, and Francis R Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In Advances in neural information processing systems, pages 2663–2671, 2012.

[5] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. Journal of Machine Learning Research, 14(Feb):567–599, 2013.

[6] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In Advances in neural information processing systems, pages 315–323, 2013.

[7] Thomas Hofmann, Aurelien Lucchi, Simon Lacoste-Julien, and Brian McWilliams. Variance reduced stochastic gradient descent with neighbors. In Advances in Neural Information Processing Systems, pages 2305–2313, 2015.

[8] Zeyuan Allen-Zhu, Yang Yuan, and Karthik Sridharan. Exploiting the structure: Stochastic gradient methods using raw clusters. In Advances in Neural Information Processing Systems, pages 1642–1650, 2016.

[9] Luo Zhi-Quan and Tseng Paul. Analysis of an approximate gradient projection method with applications to the backpropagation algorithm. Optimization Methods and Software, 4(2):85–101, 1994.

[10] OL Mangasariany and MV Solodovly. Serial and parallel backpropagation convergence via nonmonotone perturbed minimization. 1994.

[11] Dimitri P Bertsekas. Incremental least squares methods and the extended kalman filter. SIAM Journal on Optimization, 6(3):807–822, 1996.

[12] Mikhail V Solodov. Incremental gradient algorithms with stepsizes bounded away from zero. Computational Optimization and Applications, 11(1):23–35, 1998.

[13] Paul Tseng. An incremental gradient (-projection) method with momentum term and adaptive stepsize rule. SIAM Journal on Optimization, 8(2):506–531, 1998.

[14] Dimitri P Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. Optimization for Machine Learning, 2010(1-38):3, 2011.

[15] Benjamin Recht and Christopher Re. Beneath the valley of the noncommutative arithmetic-geometric mean inequality: conjectures, case-studies. Technical report, and consequences. Technical report, University of Wisconsin-Madison, 2012.
[16] Roy Frostig, Rong Ge, Sham Kakade, and Aaron Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In International Conference on Machine Learning, pages 2540–2548, 2015.

[17] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In Advances in Neural Information Processing Systems, pages 3384–3392, 2015.

[18] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. SIAM Journal on Optimization, 24(4):2057–2075, 2014.

[19] Dimitri P Bertsekas and Athena Scientific. Convex optimization algorithms. Athena Scientific Belmont, 2015.

[20] Herbert Robbins and Sutton Monro. A stochastic approximation method. The annals of mathematical statistics, pages 400–407, 1951.

[21] Léon Bottou. Online algorithms and stochastic approximations. In David Saad, editor, Online Learning and Neural Networks. Cambridge University Press, Cambridge, UK, 1998. revised, oct 2012.

[22] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. SIAM Journal on optimization, 19(4):1574–1609, 2009.

[23] Shai Shalev-Shwartz and Nathan Srebro. Svm optimization: inverse dependence on training set size. In Proceedings of the 25th international conference on Machine learning, pages 928–935. ACM, 2008.

[24] 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.

[25] Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. Combinatorica, 2(4):385–393, 1982.

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

[27] Baharan Mirzasoleiman, Amin Karbasi, Ashwinkumar Badanidiyuru, and Andreas Krause. Distributed submodular cover: Succinctly summarizing massive data. In Advances in Neural Information Processing Systems, pages 2881–2889, 2015.

[28] Baharan Mirzasoleiman, Morteza Zadimoghaddam, and Amin Karbasi. Fast distributed submodular cover: Public-private data summarization. In Advances in Neural Information Processing Systems, pages 3594–3602, 2016.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[30] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. arXiv preprint arXiv:1803.00942, 2018.

[31] Kai Lai Chung et al. On a stochastic approximation method. The Annals of Mathematical Statistics, 25(3):463–483, 1954.
A Convergence Rate Analysis

We first prove the following Lemma which is an extension of the [31], Lemma 4. 

**Lemma 3.** Let \( u_k \geq 0 \) be a sequence of real numbers. Assume there exist \( k_0 \) such that 
\[
     u_{k+1} \leq (1 - \frac{c}{k})u_k + \frac{e}{k^p} + \frac{d}{k^{p+1}}, \quad \forall k \geq k_0,
\]
where \( e > 0, d > 0, c > 0 \) are given real numbers. Then 
\[
    u_k \leq (dk^{-1} + e)(c - p + 1)^{-1}k^{-p+1} + o(k^{-p+1}) \quad \text{for } c > p - 1, p \geq 1 \quad (15)
\]
\[
    u_k = O(k^{-c} \log k) \quad \text{for } c = p - 1, p > 1 \quad (16)
\]
\[
    u_k = O(k^{-c}) \quad \text{for } c < p - 1, p > 1 \quad (17)
\]

**Proof.** Let \( c > p - 1 \) and \( v_k = k^{p-1}u_k - \frac{d}{k(c-p+1)} - \frac{e}{c-p+1} \). Then, using Taylor approximation \( (1 + \frac{t}{k})^p = (1 + \frac{p}{k}) + o(\frac{1}{k}) \) we can write 
\[
    v_{k+1} = (k+1)^{p-1}u_{k+1} - \frac{d}{(k+1)(c-p+1)} - \frac{e}{c-p+1} \quad (19)
\]
\[
    \leq k^{p-1}((1 + \frac{1}{k}))^{p-1}((1 - \frac{c}{k})u_k + \frac{e}{k^p} + \frac{d}{k^{p+1}}) - \frac{d}{(k+1)(c-p+1)} - \frac{e}{c-p+1} \quad (20)
\]
\[
    = k^{p-1}u_k\left(1 - \frac{c}{k} + 1 + \frac{e}{k^2}\right) + \frac{e}{k}(1 + \frac{p-1}{k} + o(\frac{1}{k})) \quad (21)
\]
\[
    + \frac{d}{k^2}\left(1 + \frac{1}{k} + o(\frac{1}{k})\right) - \frac{d}{k+1}\left(\frac{c}{k} + \frac{1}{k}\right) - \frac{e}{c-p+1} \quad (22)
\]
\[
    = v_k\left(1 - \frac{c-p+1}{k} + o(\frac{1}{k})\right) + \frac{d/c-p+1/k}{k(k+1)} + \frac{e(p-1)/k}{k^2} + \frac{d(p-1)/k^3}{k^3} + o(\frac{1}{k^2}) \quad (23)
\]
\[
    \leq v_k\left(1 - \frac{c-p+1}{k} + o(\frac{1}{k})\right) + \frac{d/c-p+1/k}{k(k+1)} + \frac{e(p-1)/k}{k^2} + \frac{d(p-1)/k^3}{k^3} + o(\frac{1}{k^2}) \quad (26)
\]

Note that for \( v_k \), we have 
\[
    \sum_{k=0}^{\infty} \left(1 - \frac{c-p+1}{k} + o(\frac{1}{k})\right) = \infty
\]

and 
\[
    \left(\frac{d/c-p+1/k}{k(k+1)} + \frac{e(p-1)/k}{k^2} + \frac{d(p-1)/k^3}{k^3} + o(\frac{1}{k^2})\right) \left(1 - \frac{c-p+1}{k} + o(\frac{1}{k})\right)^{-1} \to 0.
\]

Therefore, \( \lim_{k \to \infty} v_k \leq 0 \), and we get Eq. (15) For \( p = 1 \), we have \( u_k \leq \frac{e}{c} \). Hence, \( u_k \) converges into the region \( u \leq \frac{e}{c} \), with ratio \( 1 - \frac{e}{c} \).

Moreover, for \( p - 1 > c \) we have 
\[
    v_{k+1} = u_{k+1}(k+1)^c \leq \left[(1 - \frac{c}{k})u_k + \frac{e}{k^p} + \frac{d}{k^{p+1}}\right]k^c\left(1 + \frac{c}{k} + \frac{c^2}{2k^2} + o(\frac{1}{k^2})\right) \quad (27)
\]
\[
    = \left(1 - \frac{c^2}{2k^2} + o(\frac{1}{k^2})\right)v_k + \frac{d}{k^{p-c+1}}\left(1 + O(\frac{1}{k})\right) + \frac{e}{k^{p-c}}\left(1 + \frac{c}{k} + O(\frac{1}{k^2})\right) \quad (28)
\]
\[
    \leq v_k + \frac{e}{k^{p-c}} \quad (29)
\]

for sufficiently large \( k \). Summing over \( k \), we obtain that \( v_k \) is bounded for \( p - 1 > c \) (since the series \( \sum_{k=1}^{\infty}(1/k^\alpha) \) converges for \( \alpha > 1 \) and \( v_k = O(\log k) \) for \( p = c + 1 \) (since \( \sum_{i=1}^{k}(1/i) = O(\log k) \)).
In addition, based on [31, Lemma 5] for \( u_k \geq 0 \), we can write
\[
\frac{u_{k+1}}{u_k} \leq \left( 1 - \frac{c}{k^s} \right) + \frac{e}{k^p} + \frac{d}{k^q}, \quad 0 < s < 1, s \leq p < t.
\] (30)

Then, we have
\[
\frac{u_k}{u_k} \leq \frac{1}{c k^{p-s}} + o\left( \frac{1}{k^{p-s}} \right).
\] (31)

### A.1 Convergence Rate for Strongly Convex Functions

First we proof the following lemma that bounds the estimation error between \( \sum_{i \in V} f_i(x_k) \) and \( \sum_{j \in S} \gamma_j f_j(x_k) \) at the beginning of cycle \( k \). Here, we use the convention \( x_k = x_0 \).

**Lemma 4.** Assume \( f_i \)s are convex and there is a subset \( S \) of size \( r \) with corresponding per-element stepsizes \( \{ \gamma_i \} \) that estimates the full gradient by an error of at most \( \epsilon \), i.e., \( \| \sum_{j \in S} \gamma_j \nabla f_j(x) - \sum_{i \in V} \nabla f_i(x) \| \leq \epsilon \). Then we have
\[
| \sum_{j \in S} \gamma_j (f_j(x_k) - f_j(x_*)) - \sum_{i \in V} (f_i(x_k) - f_i(x_*)) | \leq \epsilon \| x_k - x_* \|. \] (32)

**Proof.** From the convexity of each component we get the following inequalities
\[
\sum_{j \in S} \gamma_j (f_j(x_k) - f_j(x_*)) \leq \| \sum_{j \in S} \gamma_j \nabla f_j(x_k) \| \cdot \| x_k - x_* \|, \] (33)
\[
\sum_{i \in V} (f_i(x_k) - f_i(x_*)) \leq \| \sum_{i \in V} \nabla f_i(x_k) \| \cdot \| x_k - x_* \| \] (34)

By subtracting the above inequalities we get
\[
| \sum_{j \in S} \gamma_j (f_j(x_k) - f_j(x_*)) - \sum_{i \in V} (f_i(x_k) - f_i(x_*)) | \leq \| \sum_{j \in S} \gamma_j \nabla f_j(x_k) \| - \| \sum_{i \in V} \nabla f_i(x_k) \| \cdot \| x_k - x_* \| \] (35)
\[
\leq \| \sum_{j \in S} \gamma_j \nabla f_j(x_k) - \sum_{i \in V} \nabla f_i(x_k) \| \cdot \| x_k - x_* \| \] (36)
\[
\leq \epsilon \| x_k - x_* \|. \] (37)
\[ \square \]

**Proof of Theorem 1**

We now provide the convergence rate for strongly convex functions building on the analysis of [1]. For non-smooth functions, gradients can be replaced by sub-gradients.

Let \( x_k = x_k \). For every IG update on subset \( S \) we have
\[
\| x_k - x_* \|^2 = \| x_{k-1} - \alpha_k \gamma_i \nabla f_i(x_{k-1}) - x_* \|^2 \] (38)
\[
\leq \| x_{k-1} - x_* \|^2 - 2 \alpha_k \gamma_i \nabla f_i(x_{k-1}) (x_{k-1} - x_*) + \alpha_k^2 \| \gamma_i \nabla f_i(x_{k-1}) \|^2 \] (39)
\[
\leq \| x_{k-1} - x_* \|^2 - 2 \alpha_k \gamma_i (f_i(x_{k-1}) - f_i(x_*)) + \alpha_k^2 \| \gamma_i \nabla f_i(x_{k-1}) \|^2. \] (40)

Adding the above inequalities over elements of \( S \) we get
\[
\| x_{k+1} - x_* \|^2 \leq \| x_k - x_* \|^2 - 2 \alpha_k \sum_{i \in S} \gamma_i (f_i(x_{k-1}) - f_i(x_*)) + \alpha_k^2 \sum_{i \in S} \| \gamma_i \nabla f_i(x_{k-1}) \|^2 \] (42)
\[
\leq \| x_k - x_* \|^2 - 2 \alpha_k \sum_{i \in S} \gamma_i (f_i(x_{k-1}) - f_i(x_*)) \] (43)
\[
+ 2 \alpha_k \sum_{i \in S} \gamma_i (f_i(x_{k-1}) - f_i(x_*)) + \alpha_k^2 \sum_{i \in S} \| \gamma_i \nabla f_i(x_{k-1}) \|^2 \] (44)
\[
(45)
where \( \gamma \)

Therefore, we get

\[
2\alpha_k \sum_{i \in S} \gamma_i (f_i(x^k_{i-1}) - f_i(x^k_i)) + \alpha_k^2 \sum_{i \in S} \|\gamma_i \nabla f_i(x^k_{i-1})\|^2
\]

From strong convexity of \( \sum_{i \in V} f_i(x) \) we get the following two inequalities

\[
\sum_{i \in V} f_i(x_k) - \sum_{i \in V} f(x_*) \geq \frac{\mu}{2} \|x_k - x_*\|^2
\]

and

\[
\|x_k - x_*\| \leq \frac{1}{\mu} \|\sum_{i \in S} \nabla f_i(x_k)\| \leq \frac{1}{\mu} (\|\sum_{i \in S} \gamma_i \nabla f_i(x_k)\| + \epsilon).
\]

Therefore,

\[
\|x_{k+1} - x_*\|^2 \leq (1 - \alpha_k \mu) \|x_k - x_*\|^2 + 2\alpha_k \epsilon (r \gamma_{\max} C + \epsilon) / \mu
\]

\[
+ 2\alpha_k \sum_{i \in S} \gamma_i (f_i(x_{i-1}) - f_i(x_k)) + \alpha_k^2 r \gamma_{\max}^2 C^2
\]

Now, from convexity we have that

\[
\gamma_i (f_i(x_k) - f_i(x^k_{i-1})) \leq \gamma_i \nabla f_i(x_k) \cdot \|x^k_{i-1} - x_k\|.
\]

In addition, we know that

\[
\|x^k_{i-1} - x_k\| \leq \alpha_k \sum_{j=1}^{i-1} \|\gamma_j \nabla f_j(x^k_{j-1})\| \leq \alpha_k(i-1) \gamma_{\max} C.
\]

Therefore, we get

\[
2\alpha_k \sum_{i \in S} \gamma_i (f_i(x_k) - f_i(x^k_{i-1})) + \alpha_k^2 r \gamma_{\max}^2 C^2 \leq 2\alpha_k \sum_{i=1}^{r} \gamma_{\max} C \cdot \alpha_k(i-1) \gamma_{\max} C + \alpha_k^2 r \gamma_{\max}^2 C^2
\]

\[
= \alpha_k^2 r^2 \gamma_{\max}^2 C^2
\]

Hence,

\[
\|x_{k+1} - x_*\|^2 \leq (1 - \alpha_k \mu) \|x_k - x_*\|^2 + 2\alpha_k \epsilon \min(D, (r \gamma_{\max} C + \epsilon) / \mu) + \alpha_k^2 r^2 \gamma_{\max}^2 C^2.
\]

where \( \gamma_{\max} \) is the size of the largest cluster, and \( C \) is the upperbound on the gradients. Note that \( \|x_k - x_*\| \) in Eq. 48 is less than the diameter of the dataset \( D \), hence we can write

\[
\|x_{k+1} - x_*\|^2 \leq (1 - \alpha_k \mu) \|x_k - x_*\|^2 + 2\alpha_k \epsilon \min(D, (r \gamma_{\max} C + \epsilon) / \mu) + \alpha_k^2 r^2 \gamma_{\max}^2 C^2.
\]

For \( 0 < s \leq 1 \), the theorem follows by applying Lemma 3 to Eq. 56 with \( c = \mu, \epsilon = 2\epsilon \min(D, (r \gamma_{\max} C + \epsilon) / \mu) \), and \( d = r^2 \gamma_{\max}^2 C^2 \).

For \( s = 0 \), where we have a constant step size \( \alpha_k = \alpha \leq \frac{1}{\mu} \), we get

\[
\|x_{k+1} - x_*\|^2 \leq (1 - \alpha \mu)^{k+1} \|x_0 - x_*\|^2 + 2\epsilon \alpha (r \gamma_{\max} C + \epsilon) \sum_{j=0}^{k} (1 - \alpha \mu)^j / \mu + \alpha^2 r^2 \gamma_{\max}^2 C^2 \sum_{j=0}^{k} (1 - \alpha \mu)^j
\]

Since \( \sum_{j=0}^{k} (1 - \alpha \mu)^j \leq \frac{1}{\alpha \mu} \), we get

\[
\|x_{k+1} - x_*\|^2 \leq (1 - \alpha \mu)^{k+1} \|x_0 - x_*\|^2 + 2\epsilon (r \gamma_{\max} C + \epsilon) / \mu^2 + \alpha r^2 \gamma_{\max}^2 C^2 / \mu,
\]

and therefore,

\[
\|x_{k+1} - x_*\|^2 \leq (1 - \alpha \mu)^{k+1} \|x_k - x_*\|^2 + 2\epsilon \min(D, (r \gamma_{\max} C + \epsilon) / \mu^2) + \alpha r^2 \gamma_{\max}^2 C^2 / \mu.
\]
A.2 Convergence Strongly Convex and Smooth Component Functions

Proof of Theorem\ref{thm:convergence}

IG updates for cycle $k$ on subset $S$ can be written as

\begin{equation}
\label{eq:ig_update}
x_{k+1} = x_k - \alpha_k (\nabla f_i(x_k) - e_k)
\end{equation}

\begin{equation}
\label{eq:epsilon_update}
e_k = \sum_{i \in S} \gamma_i (\nabla f_i(x_k) - \nabla f_i(x^k_{i-1}))
\end{equation}

Building on the analysis of \cite{2}, for convex and twice continuously differentiable function, we can write

\begin{equation}
\label{eq:component_function}
\sum_{i \in S} \gamma_i \nabla f_i(x_k) = A_k(x_k - x^*)
\end{equation}

where $A_k^r = \int_0^1 \nabla^2 f(x + \tau(x_k - x^*))d\tau$ is average of the Hessian matrices corresponding to the $r$ (weighted) elements of $S$ on the interval $[x_k, x^*]$. From Eq.\ref{eq:component_function} we have

\begin{equation}
\label{eq:weighted_gradient}
\sum_{i \in V} \nabla f_i(x_k) - \sum_{i \in S} \gamma_i \nabla f_i(x_k) = A_k(x_k - x^*) - A_k^r(x_k - x^*)
\end{equation}

where $A_k$ is average of the Hessian matrices corresponding to all the $n$ component functions on the interval $[x_k, x^*]$. Taking norm of both sides, we get

\begin{equation}
\label{eq:weighted_gradient_bound}
\| (A_k - A_k^r)(x_k - x^*) \| = \| \sum_{i \in V} \nabla f_i(x_k) - \sum_{i \in S} \gamma_i \nabla f_i(x_k) \| \leq \epsilon,
\end{equation}

where $\epsilon$ is the estimation error of the full gradient by the weighted gradients of the elements of the subset $S$.

Substituting Eq.\ref{eq:weighted_gradient_bound} into Eq.\ref{eq:ig_update} we obtain

\begin{equation}
\label{eq:ig_update_bound}
x_{k+1} - x^* = (I - \alpha_k A_k^r)(x_k - x^*) + \alpha_k e_k
\end{equation}

Taking norms of both sides, we get

\begin{equation}
\label{eq:ig_bound}
\| x_{k+1} - x^* \| \leq \| I - \alpha_k A_k^r \| \| x_k - x^* \| + \alpha_k \| e_k \|
\end{equation}

Now, we have

\begin{align}
\| (I - \alpha_k A_k^r)(x_k - x^*) \| &= \| I(x_k - x^*) - \alpha_k A_k^r(x_k - x^*) \| \label{eq:ig_bound_1} \\
&= \| (x_k - x^*) - \alpha_k (A_k^r - A_k)(x_k - x^*) - \alpha_k A_k(x_k - x^*) \| \label{eq:ig_bound_2} \\
&\leq \| (I - \alpha_k A_k)(x_k - x^*) \| + \alpha_k \| (A_k - A_k^r)(x_k - x^*) \| \label{eq:ig_bound_3} \\
&\leq \| (I - \alpha_k A_k)(x_k - x^*) \| + \alpha_k \epsilon \label{eq:ig_bound_4}
\end{align}

Substituting into Eq.\ref{eq:ig_bound} we obtain

\begin{equation}
\label{eq:ig_bound_5}
\| x_{k+1} - x^* \| \leq \| I - \alpha_k A_k \| \cdot \| x_k - x^* \| + \alpha_k \epsilon + \alpha_k \| e_k \|
\end{equation}

From strong convexity of $\sum_{i \in V} f_i(x)$, and gradient smoothness of each component $f_i(x)$ we have

\begin{equation}
\label{eq:strong_convexity}
\mu I_n \preceq \sum_{i \in V} \nabla^2 f_i(x), A_k \preceq \beta I_n, \quad x \in \mathcal{X},
\end{equation}

where $\beta = \sum_{i \in V} \beta_i$. In addition, from the gradient smoothness of the components we can write

\begin{align}
\| e_k \| &\leq \sum_{i=1}^r \gamma_i \beta_i \| x_k - x^k_i \| \label{eq:gradient_smoothness_1} \\
&\leq \sum_{i=1}^r \gamma_i \beta_i \sum_{j=1}^{i-1} \| x^k_j - x^k_i \| \label{eq:gradient_smoothness_2} \\
&\leq \sum_{i=1}^r \gamma_i \beta_i \alpha_k \sum_{j=1}^{i-1} \| \gamma_j \nabla f_j(x^k_j) \| \label{eq:gradient_smoothness_3} \\
&\leq \alpha_k \beta C \gamma_{\max}^2 \label{eq:gradient_smoothness_4}
\end{align}
Therefore,
\[ \| x_{k+1} - x^* \| \leq \max(\| 1 - \alpha_k \mu \|, \| 1 - \alpha_k \beta \|) \| x_k - x^* \| + \alpha_k \epsilon + \alpha_k^2 \beta C r_\gamma^2 \] (79)
\[ \leq (1 - \alpha_k \mu) \| x_k - x^* \| + \alpha_k \epsilon + \alpha_k^2 \beta C r_\gamma^2 \] if \( \alpha_k \beta \leq 1. \) (80)

For \( 0 < s \leq 1, \) the theorem follows by applying Lemma 3 to Eq. 79 with \( c = \mu, \epsilon = \epsilon, d = \beta C r_\gamma^2 \) for classification. For regularized logistic regression with ridge regression, we also need some additional steps. For classification, we require \( y_i = y_j \), hence we can select subsets from each class and then merge the results. On the other hand, in ridge regression we also need \( |y_i - y_j| \) to be small. Similar results can be deduced for other loss functions including square loss, smoothed hinge loss, etc.

B Norm of the Difference Between Gradients

For ridge regression \( f_i(x) = \frac{1}{2}(\langle a_i, x \rangle - y_i)^2 + \frac{1}{2} \| x \|^2, \) we have \( \nabla f_i(x) = \langle a_i, x \rangle - y_i + \lambda x. \) Therefore,
\[ \| \nabla f_i(x) - \nabla f_j(x) \| = (\| a_i - a_j \| \cdot \| x \| + \| y_i - y_j \|) \| a_j \| \] (84)

For \( \| a_i \| \leq 1, \) and \( y_i = y_j \) we get
\[ \| \nabla f_i(x) - \nabla f_j(x) \| \leq \| a_i - a_j \| O(\| x \|) \] (85)

For regularized logistic regression with \( y \in \{-1, 1\}, \) we have \( \nabla f_i(x) = y_i/(1 + e^{y_i\langle a_i, x \rangle}). \) For \( y_i = y_j \) we get
\[ \| \nabla f_i(x) - \nabla f_j(x) \| = \frac{e^{\| a_i - a_j \| \cdot \| x \|} - 1}{1 + e^{\langle a_i, x \rangle}} \| a_j \|. \] (86)

For \( \| a_i \| \leq 1, \) using Taylor approximation \( e^x \leq 1 + x, \) and noting that \( \frac{1}{1 + e^{-a_i \cdot x}} \leq 1 \) we get
\[ \| \nabla f_i(x) - \nabla f_j(x) \| \leq \frac{\| a_i - a_j \| \cdot \| x \|}{1 + e^{-\langle a_i, x \rangle}} \| a_j \| \leq \| a_i - a_j \| O(\| x \|). \] (87)

For classification, we require \( y_i = y_j, \) hence we can select subsets from each class and then merge the results. On the other hand, in ridge regression we also need \( |y_i - y_j| \) to be small. Similar results can be deduced for other loss functions including square loss, smoothed hinge loss, etc.

Assuming \( \| x \| \) is bounded for all \( x \in \mathcal{X}, \) upper-bounds on the euclidean distances between the gradients can be pre-computed.