Abstract
Incremental gradient (IG) methods, such as stochastic gradient descent and its variants are commonly used for large scale optimization in machine learning. Despite the sustained effort to make IG methods more data-efficient, it remains an open question how to select a training data subset that can theoretically and practically perform on par with the full dataset. Here we develop CRAIG, a method to select a weighted subset (or coreset) of training data that closely estimates the full gradient by maximizing a submodular function. We prove that applying IG to this subset is guaranteed to converge to the (near)optimal solution with the same convergence rate as that of IG for convex optimization. As a result, CRAIG achieves a speedup that is inversely proportional to the size of the subset. To our knowledge, this is the first rigorous method for data-efficient training of general machine learning models. Our extensive set of experiments show that CRAIG, while achieving practically the same solution, speeds up various IG methods by up to 6x for logistic regression and 3x for training deep neural networks.

1. Introduction
Mathematical optimization lies at the core of training large-scale machine learning systems, and is now widely used over massive data sets with great practical success, assuming sufficient data resources are available. Achieving this success, however, also requires large amounts of (often GPU) computing, as well as concomitant financial expenditures and energy usage (Strubell et al., 2019). Significantly decreasing these costs without decreasing the learnt system’s resulting accuracy is one of the grand challenges of machine learning and artificial intelligence today (Asi & Duchi, 2019).

Training machine learning models often reduces to optimizing a regularized empirical risk function. Given a convex loss $l$, and a $\mu$-strongly convex regularizer $r$, one aims to find model parameter vector $w_*$ over the parameter space $\mathcal{W}$ that minimizes the loss $f$ over the training data $V$:

$$\begin{align*}
    w_* &\in \arg\min_{w \in \mathcal{W}} f(w), \quad f(w) := \sum_{i \in V} f_i(w) + r(w), \\
    f_i(w) &= l(w, (x_i, y_i)),
\end{align*}$$

where $V = \{1, \ldots, n\}$ is an index set of the training data, and functions $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$ are associated with training examples $(x_i, y_i)$, where $x_i \in \mathbb{R}^d$ is the feature vector, and $y_i$ is the point $i$’s label.

Standard Gradient Descent can find the minimizer of this problem, but requires repeated computations of the full gradient $\nabla f(w)$—sum of the gradients over all training data points/functions $i$—and is therefore prohibitive for massive data sets. This issue is further exacerbated in case of deep neural networks where gradient computations (backpropagation) are expensive. Incremental Gradient (IG) methods, such as Stochastic Gradient Descent (SGD) and its accelerated variants, including SGD with momentum (Qian, 1999), Adagrad (Duchi et al., 2011), Adam (Kingma & Ba, 2014), SAGA (Defazio et al., 2014), and SVRG (Johnson & Zhang, 2013) iteratively estimate the gradient on random subsets/batches of training data. While this provides an unbiased estimate of the full gradient, the randomized batches introduce variance in the gradient estimate (Hofmann et al., 2015), and therefore stochastic gradient methods are in general slow to converge (Johnson & Zhang, 2013; Defazio et al., 2014). The majority of the work speeding up IG methods has thus primarily focused on reducing the variance of the gradient estimate (SAGA (Defazio et al., 2014), SVRG (Johnson & Zhang, 2013), Katysha (Allen-Zhu, 2017)) or more carefully selecting the gradient stepsize (Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), Adam (Kingma & Ba, 2014)).

However, the direction that remains largely unexplored is how to carefully select a small subset $S \subseteq V$ of the full training data $V$, so that the model is trained only on the subset $S$ while still (approximately) converging to the globally optimal solution (i.e., the model parameters that would be obtained if training/optimizing on the full $V$). If such
a subset $S$ can be quickly found, then this would directly lead to a speedup of $|V|/|S|$ (which can be very large if $|S| \ll |V|$) per epoch of IG.

There are four main challenges in finding such a subset $S$. First is that a guiding principle for selecting $S$ is unclear. For example, selecting training points close to the decision boundary might allow the model to fine tune the decision boundary, while picking the most diverse set of data points would allow the model to get a better sense of the training data distribution. Second is that finding $S$ must be fast, as otherwise identifying the set $S$ may take longer than the actual optimization, and so no overall speed-up would be achieved. Third is that finding a subset $S$ is not enough. One also has to decide on a gradient stepsize for each data point in $S$, as they affect the convergence. And last, while the method might work well empirically on some data sets, one also requires theoretical understanding and mathematical convergence guarantees.

Here we develop Coresets for Accelerating Incremental Gradient descent (CRAIG), for selecting a subset of training data points to speed up training of large machine learning models. Our key idea is to select a weighted subset $S$ of training data $V$ that best approximates the full gradient of $V$. We prove that the subset $S$ that minimizes an upper-bound on the error of estimating the full gradient maximizes a submodular facility location function. Hence, $S$ can be efficiently found using a fast greedy algorithm.

We also provide theoretical analysis of CRAIG and prove its convergence. Most importantly, we show that any incremental gradient method (IG) on $S$ converges in the same number of epochs as the same IG would on the full $V$, which means that we obtain a speed-up inversely proportional to the size of $S$. In particular, for a $\mu$-strongly convex risk function and a subset $S$ selected by CRAIG that estimates the full gradient by an error of at most $\epsilon$, we prove that IG on $S$ with diminishing stepsize $\alpha_k = \alpha/k^\tau$ at epoch $k$ (with $0 < \tau < 1$ and $0 < \alpha$), converges to an $2R\epsilon/\mu^2$ neighborhood of the optimal solution at rate $O(1/\sqrt{k})$. Here, $R = \min\{d_0, (r_{\gamma \text{max}} C + \epsilon)/\mu\}$ where $d_0$ is the initial distance to the optimum, $C$ is an upper-bound on the norm of the gradients, $r = |S|$, and $\gamma_{\text{max}}$ is the largest weight for the elements in the subset obtained by CRAIG. Moreover, we prove that if in addition to the strong convexity, component functions have smooth gradients, IG with the same diminishing step size on subset $S$ converges to a $2\epsilon/\mu$ neighborhood of the optimum solution at rate $O(1/k^\tau)$.

The above implies that IG on $S$ converges to the same solution and in the same number of epochs as IG on the full $V$. But because every epoch only uses a subset $S$ of the data, it requires fewer gradient computations and thus leads to a $|V|/|S|$ speedup over traditional IG methods, while still (approximately) converging to the optimal solution. We also note that CRAIG is complementary to various incremental gradient (IG) methods (SGD, SAGA, SVRG, Adam), and such methods can be used on the subset $S$ found by CRAIG.

We also demonstrate the effectiveness of CRAIG via an extensive set of experiments using logistic regression (a convex optimization problem) as well as training deep neural networks (non-convex optimization problems). We show that CRAIG speeds up incremental gradient methods, including SGD, SAGA, and SVRG. In particular, CRAIG while achieving practically the same loss and accuracy as the underlying incremental gradient descent methods, speeds up gradient methods by up to 6x for convex and 3x for non-convex loss functions.

2. Related Work

Convergence of IG methods has been long studied under various conditions (Zhi-Quan & Paul, 1994; Mangasarian & Solodov, 1994; Bertsekas, 1996; Solodov, 1998; Tseng, 1998), however IG’s convergence rate has been characterized only more recently (see Bertsekas, 2015b) for a survey). In particular, (Nedić & Bertsekas, 2001) provides a $O(1/\sqrt{K})$ convergence rate for diminishing stepsizes $\alpha_k$ per epoch $k$ under a strong convexity assumption, and (Gürbüzbalaban et al., 2015) proves a $O(1/k^\tau)$ convergence rate with diminishing stepsizes $\alpha_k = \Theta(1/k^\tau)$ for $\tau \in (0, 1]$ under an additional smoothness assumption for the components. While these works provide convergence on the full dataset, our analysis provides the same convergence rates on subsets obtained by CRAIG.

Techniques for speeding up SGD, are mostly focused on variance reduction techniques (Roux et al., 2012; Shalev-Shwartz & Zhang, 2013; Johnson & Zhang, 2013; Hofmann et al., 2015; Allen-Zhu et al., 2016), and accelerated gradient methods when the regularization parameter is small (Frostig et al., 2015; Lin et al., 2015; Xiao & Zhang, 2014). Very recently, (Hofmann et al., 2015; Allen-Zhu et al., 2016) exploited neighborhood structure to further reduce the variance of stochastic gradient descent and improve its running time. Our CRAIG method and analysis are complementary to variance reduction and accelerated methods. CRAIG can be applied to all these methods as well to speed them up.

Coresets are weighted subsets of the data, which guarantee that models fitting the coreset also provide a good fit for the original data. Coreset construction methods traditionally perform importance sampling with respect to sensitivity score, to provide high-probability solutions (Har-Peled & Mazumdar, 2004; Lucic et al., 2017; Cohen et al., 2017) for a particular problem, such as $k$-means and $k$-median clustering (Har-Peled & Mazumdar, 2004), naïve Bayes and nearest-neighbors (Wei et al., 2015), mixture mod-
els (Lucic et al., 2017), low rank approximation (Cohen et al., 2017), spectral approximation (Agarwal et al., 2004; Li et al., 2013), Nyström methods (Agarwal et al., 2004; Musco & Musco, 2017), and Bayesian inference (Campbell & Broderick, 2018). Unlike existing coreset construction algorithms, our method is not problem specific and can be applied for training general machine learning models.

3. Coresets for Accelerating Incremental Gradient Descent (CRAIG)

We proceed as follows: First, we define an objective function $L$ for selecting an optimal set $S$ of size $r$ that best approximates the gradient of the full training dataset $V$ of size $n$. Then, we show that $L$ can be turned into a submodular function $F$ and thus $S$ can be efficiently found using a fast greedy algorithm. Crucially, we also show that for convex loss functions the approximation error between the estimated and the true gradient can be efficiently minimized in a way that is independent of the actual optimization procedure. Thus, CRAIG can simply be used as a preprocessing step before the actual optimization starts.

Incremental gradient methods aim at estimating the full gradient $\nabla f(w)$ over $V$ by iteratively making a step based on the gradient of every function $f_i$. Our key idea in CRAIG is that if we can find a small subset $S$ such that the weighted sum of the gradients of its elements closely approximates the full gradient over $V$, we can apply IG only to the set $S$ (with stepsizes equal to the weight of the elements in $S$), and we should still converge to the (approximately) optimal solution, but much faster.

Specifically, our goal in CRAIG is to find the smallest subset $S \subseteq V$ and corresponding per-element stepsizes $\gamma_j \geq 0$ that approximate the full gradient with an error at most $\epsilon > 0$ for all the possible values of the optimization parameters $w \in W$.

$$S^* = \arg \min_{S \subseteq V, \gamma_j \geq 0} \| \gamma_j \nabla f_j(w) \|, \quad \text{s.t.} \quad \max_{w \in W} \left\| \sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \nabla f_j(w) \right\| \leq \epsilon.$$  

(2)

Given such an $S^*$ and associated weights $\{\gamma_j\}$, we are guaranteed that gradient updates on $S^*$ will be similar to the gradient updates on $V$ regardless of the value of $w$.

Unfortunately, directly solving the above optimization problem is not feasible, due to two problems. Problem 1: Eq. (2) requires us to calculate the gradient of all the functions $f_i$ over the entire space $W$, which is too expensive and would not lead to overall speedup. In other words, it would appear that solving for $S^*$ is as difficult as solving Eq. (1), as it involves calculating $\sum_{i \in V} \nabla f_i(w)$ for various $w \in W$.

And Problem 2: even if calculating the normed difference between the gradients in Eq. (2) would be fast, as we discuss later finding the optimal subset $S^*$ in NP-hard. In the following, we address the above two challenges and discuss how we can quickly find a near-optimal subset $S$.

3.1. Upper-bound on the Estimation Error

We first address Problem 1, i.e., how to quickly estimate the error/discrepancy of the weighted sum of gradients of functions $f_j$ associate with data points $j \in S$, vs the full gradient, for every $w \in W$.

Let $S$ be a subset of $r$ data points. Furthermore, assume that there is a mapping $\varsigma_w : V \to S$ that for every $w \in W$ assigns every data point $i \in V$ to one of the elements $j \in S$. Let $C_j = \{i \in [n] | \varsigma(i) = j \} \subseteq V$ be the set of data points that are assigned to $j \in S$, and $\gamma_j = |C_j|$ be the number of such data points. Hence, $\{C_j\}_{j=1}^r$ form a partition of $V$. Then, for any arbitrary (single) $w \in W$ we can write

$$\sum_{i \in V} \nabla f_i(w) = \sum_{i \in V} \left( \nabla f_i(w) - \nabla f_{\varsigma(i)}(w) + \nabla f_{\varsigma(i)}(w) \right)$$

$$= \sum_{i \in V} \left( \nabla f_i(w) - \nabla f_{\varsigma(i)}(w) \right) + \sum_{j \in S} \gamma_j \nabla f_j(w).$$  

(3)

(4)

Subtracting and then taking the norm of the both sides, we get an upper bound on the error of estimating the full gradient with the weighted sum of the gradients of the functions $f_j$ for $j \in S$. I.e.,

$$\| \sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \nabla f_j(w) \| \leq \sum_{i \in V} \| \nabla f_i(w) - \nabla f_{\varsigma(i)}(w) \|.$$  

(5)

where the inequality follows from the triangle inequality. The upper-bound in Eq. (5) is minimized when $\varsigma_w$ assigns every $i \in V$ to an element in $S$ with most gradient similarity at $w$, or minimum Euclidean distance between the gradient vectors at $w$. That is: $\varsigma_w(i) \in \arg \min_{j \in S} \| \nabla f_i(w) - \nabla f_j(w) \|$. Hence,

$$\min_{S \subseteq V} \left\| \sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \nabla f_j(w) \right\| \leq \sum_{i \in V} \min_{j \in S} \| \nabla f_i(w) - \nabla f_j(w) \|.$$  

(6)

The right hand side of Eq. (6) is minimized when $S$ is the set of $r$ medoids (exemplars) (Kaufman et al., 1987) for all the components in the gradient space.
So far, we considered upper-bounding the gradient estimation error at a particular \( w \in \mathcal{W} \). To bound the estimation error for all \( w \in \mathcal{W} \), we consider a worst-case approximation of the estimation error over the entire parameter space \( \mathcal{W} \). Formally, we define a distance metric \( d_{ij} \) between gradients of \( f_i \) and \( f_j \) as the maximum normed difference between \( \nabla f_i(w) \) and \( \nabla f_j(w) \) over all \( w \in \mathcal{W} \):

\[
d_{ij} = \max_{w \in \mathcal{W}} ||\nabla f_i(w) - \nabla f_j(w)||. \tag{7}
\]

Thus, by solving the following minimization problem, we obtain the smallest weighted subset \( S^* \) that approximates the full gradient by an error of at most \( \epsilon \) for all \( w \in \mathcal{W} \):

\[
S^* = \arg\min_{S \subseteq \mathcal{V}} |S|, \quad \text{s.t.} \quad L(S) = \sum_{i \in V} \min_{j \in S} d_{ij} \leq \epsilon. \tag{8}
\]

Note that Eq. (8) requires that the gradient error is bounded over \( \mathcal{W} \). However, we show (Appendix B.1) for several classes of convex problems, including linear regression, ridge regression, logistic regression, and regularized support vector machines (SVMs), the normed gradient difference between data points can be efficiently bounded approximately by (Allen-Zhu et al., 2016; Hofmann et al., 2015):

\[
\forall w, i, j, \quad \|\nabla f_i(w) - \nabla f_j(w)\| \leq d_{ij} \leq \max_{w \in \mathcal{W}} \mathcal{O}(\|w\|) \cdot \|x_i - x_j\| = \text{const.} \|x_i - x_j\|. \tag{9}
\]

Note when \( \|w\| \) is bounded for all \( w \in \mathcal{W} \), i.e., \( \max_{w \in \mathcal{W}} \mathcal{O}(\|w\|) < \infty \), upper-bounds on the Euclidean distances between the gradients can be pre-computed. This is crucial, because it means that estimation error of the full gradient can be efficiently bounded independent of the actual optimization problem (i.e., \( \mathcal{W} \)). Thus, these upper-bounds can be computed only once as a pre-processing step before any training takes place, and then used to find the subset \( S \) by solving the optimization problem (8). We address upper-bounding the normed difference between gradients for deep models in Section 3.4.1.

### 3.2. The CRAIG Algorithm

Optimization problem (8) produces a subset \( S \) of elements with their associated weights \( \gamma_j \) or per-element stepsizes that closely approximates the full gradient. Here, we show how to efficiently approximately solve the above optimization problem to find a near-optimal subset \( S \).

The optimization problem (8) is NP-hard as it involves calculating the value of \( L(S) \) for all the \( 2^{\mathcal{V}} \) subsets \( S \subseteq \mathcal{V} \). We show, however, that we can transform it into a submodular set cover problem, that can be efficiently approximated.

Formally, \( F \) is submodular if \( F(S \cup \{e\}) - F(S) \geq F(T \cup \{e\}) - F(T) \), for any \( S \subseteq T \subseteq \mathcal{V} \) and \( e \in \mathcal{V} \setminus T \). We denote the marginal utility of an element \( s \) w.r.t. a subset \( S \) as \( F(e|S) = F(S \cup \{e\}) - F(S) \). Function \( F \) is called monotone if \( F(e|S) \geq 0 \) for any \( e \in \mathcal{V} \setminus S \) and \( S \subseteq \mathcal{V} \). The submodular cover problem is defined as finding the smallest set \( S \) that achieves utility \( \rho \). Precisely,

\[
S^* = \arg\min_{S \subseteq \mathcal{V}} |S|, \quad \text{s.t.} \quad F(S) \geq \rho. \tag{10}
\]

Although finding \( S^* \) is NP-hard since it captures such well-known NP-hard problems such as Minimum Vertex Cover, for many classes of submodular functions, a simple greedy algorithm is known to be very effective (Nemhauser et al., 1978; Wolsey, 1982). The greedy algorithm starts with the empty set \( S_0 = \emptyset \), and at each iteration \( i \), it chooses an element \( e \in \mathcal{V} \) that maximizes \( F(e|S_{i-1}) \), i.e., \( S_i = S_{i-1} \cup \{\arg\max_{e \in \mathcal{V}} F(e|S_{i-1})\} \). Greedy gives us a logarithmic approximation, i.e. \( |S| \leq (1 + \ln(\max\{F(e|\emptyset)\}))|S^*| \) (Wolsey, 1982). The computational complexity of the greedy algorithm is \( \mathcal{O}(|\mathcal{V}| \cdot |S|) \). However, its running time can be reduced to \( \mathcal{O}(|\mathcal{V}|) \) using stochastic algorithms (Mirzasoleiman et al., 2015a) and further improved using lazy evaluation (Minoux, 1978), and distributed implementations (Mirzasoleiman et al., 2015b; 2016). Given a subset \( S \subseteq \mathcal{V} \), the facility location function quantifies the coverage of the whole data set \( \mathcal{V} \) by the subset \( S \) by summing the similarities between every \( i \in \mathcal{V} \) and its closest element \( j \in S \). Formally, facility location is defined as \( F_{fl}(S) = \sum_{i \in \mathcal{V}} \max_{j \in S} s_{i,j} \), where \( s_{i,j} \) is the similarity between \( i, j \in \mathcal{V} \). The facility location function has been used in various summarization applications (Lin et al., 2009; Lin & Bilmes, 2012). By introducing an auxiliary element \( s_0 \) we can turn \( L(S) \) in Eq. (8) into a monotone submodular facility location function,

\[
F(S) = L(S) - L(S \cup \{s_0\}), \tag{11}
\]

where \( L(S) \) 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. For a 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 Eq. (8):

\[
S^* = \arg\min_{S \subseteq \mathcal{V}} |S|, \quad \text{s.t.} \quad F(S) \geq L(S) - \epsilon. \tag{12}
\]

At every step, the greedy algorithm selects an element that reduces the upper bound on the estimation error the most. In fact, the size of the smallest subset \( S \) that estimates the full gradient by an error of at most \( \epsilon \) depends on the structural properties of the data. Intuitively, 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, and are used as stepsize of element \( j \in S \) during IG. The pseudocode for CRAIG is outlined in Algorithm 1.
Algorithm 1 CRAIG (CoResets for Accelerating Incremental Gradient descent)

**Input:** Set of component functions \( f_i \) for \( i \in V = [n] \).

**Output:** Subset \( S \subseteq V \) with corresponding per-element stepsizes \( \{\gamma_i\}_{i \in S} \).

1. \( S_0 \leftarrow \emptyset, s_0 = 0, i = 0 \).
2. **while** \( F(S) < L(\{s_0\}) - \epsilon \) **do**
3. \( j \in \arg \max_{e \in V \backslash S_{i-1}} F(e|S_{i-1}) \)
4. \( S_i = S_{i-1} \cup \{j\} \)
5. \( i = i + 1 \)
6. **end while**
7. **for** \( j = 1 \) to \(|S|\) **do**
8. \( \gamma_j = \sum_{i \in V} \| j = \arg \min_{s \in S} \max_{w \in W} \| \nabla f_i(w) - \nabla f_i(w) \| \|
9. **end for**

Notice that CRAIG creates subset \( S \) incrementally one element at a time, which produces a natural order to the elements in \( S \). Adding the element with largest marginal gain \( j \in \arg \max_{e \in V} F(e|S_{i-1}) \) improves our estimation from the full gradient by an amount bounded by the marginal gain. At every step \( i \), we have \( F(S_i) \geq (1 - e^{-i/|S|})F(S^*) \). Hence, for a greedily ordered subset \( S = \{s_1, \ldots, s_k\} \), we have
\[
\left\| \sum_{i \in V} \nabla f_i(w) - \sum_{j=1}^{k} \gamma_j \nabla f_i(w) \right\| \leq c - \left(1 - e^{-j/k}\right)L(S^*),
\]
(13)
where \( c \) is a constant. Intuitively, the first elements of the ordering contribute the most to provide a close approximation of the full gradient and the rest of the elements further refine the approximation. Hence, the first incremental gradient updates gets us close to \( w^* \), and the rest of the updates further refine the solution.

3.3. CRAIG with Limited Budget

In practice, we often have a limited budget in terms of time or computational resources, and we are interested to find a near-optimal subset of size \( r \) that best approximates the full gradient. This problem can be formulated as a submodular maximization problem which is dual to the submodular cover problem (12):
\[
S^* = \arg \max_{S \subseteq V} F(S), \quad \text{s.t.} \quad |S| \leq r.
\]
(14)
For the above submodular maximization problem, the greedy algorithm discussed in Section 3.2 provides a \((1 - 1/e)\)-approximation to the optimal solution. For a subset \( S \) of size at most \( r \) obtained by the greedy algorithm, we can calculate the value of \( \epsilon \) as follows:
\[
\epsilon \leq F(S) - L(\{s_0\}).
\]
(15)
We use this formulation in our experiments in Section 5.

3.4. Application of CRAIG to Deep Networks

As discussed, CRAIG selects a subset that closely approximates the full gradient, and hence can be also applied for speeding up training deep networks. The challenge here is that we cannot use inequality (9) to bound the normed difference between gradients for all \( w \in W \) and find the subset as a preprocessing step.

However, for deep neural networks, the variation of the gradient norms is mostly captured by the gradient of the loss w.r.t. the input to the last layer [Section 3.2 of (Katharopoulos & Fleuret, 2018)]. We show (Appendix B.1) that the normed gradient difference between data points can be efficiently bounded approximately by
\[
\|\nabla f_i(w) - \nabla f_j(w)\| \leq c_1 \|\Sigma_L(z_i^{(L)}) \nabla f_i^{(L)}(w) - \Sigma_L(z_j^{(L)}) \nabla f_j^{(L)}(w)\| + c_2,
\]
where \( \Sigma_L(z_i^{(L)}) \nabla f_i^{(L)}(w) \) is gradient of the loss w.r.t. the input to the last layer for data point \( i \), and \( c_1, c_2 \) are constants. The above upper-bound depends on parameter vector \( w \) which changes during the training process. Thus, we need to use CRAIG to update the subset \( S \) after a number of parameter updates.

The above upper-bound is often only slightly more expensive than calculating the loss. For example, in cases where we have cross entropy loss with soft-max as the last layer, the gradient of the loss w.r.t. the \( i \)-th input to the soft-max is simply \( p_i - y_i \), where \( p_i \) are logits (dimension \( p-1 \) for \( p \) classes) and \( y \) is the one-hot encoded label. In this case, CRAIG does not need any backward pass or extra storage. Note that, although CRAIG needs an additional \( O(|V| \cdot |S|) \) complexity (or \( O(|V|) \) using stochastic greedy) to find the subset \( S \) at the beginning of every epoch, this complexity does not involve any (exact) gradient calculations and is negligible compared to the cost of backpropagations performed during the epoch. Hence, as we show in the experiments CRAIG is practical and scalable.

4. Convergence Rate Analysis of CRAIG

The idea of CRAIG is to selects a subset that closely approximates the full gradient, and hence can be applied to speed up most IG variants as we show in our experiments. Here, we briefly introduce the original IG method, and then prove the convergence rate of IG applied to CRAIG subsets.

4.1. Incremental Gradient Methods (IG)

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
where $w_0^k \in \mathbb{R}^d$, it makes $k$ passes over all the $n$ components. At every epoch $k \geq 1$, it iteratively updates $w_{i-1}^k$ based on the gradient of $f_i$ for $i = 1, \ldots, n$ using stepsize $\alpha_k > 0$. I.e.,

$$w_i^k = w_{i-1}^k - \alpha_k \nabla f_i(w_{i-1}^k), \quad i = 1, 2, \ldots, n, \quad (17)$$

with the convention that $w_0^{k+1} = w_0^k$. Note that for a closed and convex subset $W$ of $\mathbb{R}^d$, the results can be projected onto $W$, and the update rule becomes

$$w_i^k = P_W(w_{i-1}^k - \alpha_k \nabla f_i(w_{i-1}^k)), \quad i = 1, 2, \ldots, n, \quad (18)$$

where $P_W$ denotes projection on the set $W \subset \mathbb{R}^d$.

IG with diminishing stepsizes converges at rate $O(1/\sqrt{k})$ for strongly convex sum function (Nedić & Bertsekas, 2001). If in addition to the strong convexity of the sum function, every component function $f_i$ is smooth, IG with diminishing stepsizes $\alpha_k = \Theta(1/k^s)$, $s \in (0, 1]$ converges at rate $O(1/k^s)$ (Gürbüzbalaban et al., 2015).

The convergence rate analysis of IG is valid regardless of the order of processing the elements. However, in practice, the convergence rate of IG is known to be quite sensitive to the order of processing the elements (Bertsekas, 2015a; Gürbüzbalaban et al., 2017). If problem-specific knowledge can be used to find a favorable order $\sigma$ (defined as a permutation $\{\sigma_1, \ldots, \sigma_n\}$ of $\{1, 2, \ldots, n\}$), IG can be updated to process the functions according to this order, i.e.,

$$w_i^k = w_{i-1}^k - \alpha_k \nabla f_{\sigma_i}(w_{i-1}^k), \quad i = 1, 2, \ldots, n. \quad (19)$$

In general a favorable order is not known in advance. A common approach is sampling the function indices with replacement from the set $\{1, 2, \ldots, n\}$, which is called the Stochastic Gradient Descent (SGD) method.

### 4.2. Convergence Rate of IG on CRAIG Subsets

Next we analyze the convergence rate of IG applied to the weighted subset $S$ found by CRAIG. Note that CRAIG finds $S$ by greedily minimizing (12) (or maximizing (14)). Therefore, $S$ is a near-optimal solution of problem (2) and estimates the full gradient by an error of at most $\epsilon$, i.e.,

$$\max_{w \in W} \|\sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \nabla f_j(w)\| \leq \epsilon.$$

Here, we show that (1) applying IG to $S$ converges to a close neighborhood of the optimal solution and that (2) this convergence happens at the same rate (same number of epochs) as IG on the full data. Formally, every step of IG on the subset becomes

$$w_i^k = w_{i-1}^k - \alpha_k \gamma_{s_i} \nabla f_{s_i}(w_{i-1}^k), \quad i = 1, 2, \ldots, r, \quad s_i \in S, \quad |S| = r. \quad (20)$$

Here, $\sigma$ is a permutation of $\{1, 2, \ldots, r\}$, and the per-element stepsize $\gamma_{s_i}$ for every function $f_{s_i}$ is the weight of the element $s_i \in S$ and is fixed for all epochs.

### 4.3. Convergence for Strongly Convex $f$

We first provide the convergence analysis for the case where the function $f$ in Problem (1) is strongly convex, i.e. $\forall w, w' \in \mathbb{R}^d$ we have $f(w) \geq f(w') + \langle \nabla f(w'), w - w' \rangle + \frac{1}{2} \|w - w'\|^2$.

**Theorem 1.** Assume that $f$ is strongly convex, and $S$ is a weighted subset of size $r$ obtained by CRAIG that estimates the full gradient by an error of at most $\epsilon$, i.e.,

$$\max_{w \in W} \|\sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \nabla f_j(w)\| \leq \epsilon.$$

Then for the iterates $\{w_k = w_0^k\}$ generated by applying IG to $S$ with per-epoch stepsize $\alpha_k = \alpha/k^r$ with $\alpha > 0$ and $\tau \in [0, 1]$, we have

(i) if $\tau = 1$, then $\|w_k - w_*\|^2 \leq 2\epsilon R/\mu^2 + \alpha^2 \gamma_{\text{max}}^2 C^2/\kappa\mu$,

(ii) if $0 < \tau < 1$, then $\|w_k - w_*\|^2 \leq 2\epsilon R/\mu^2$, for $k \to \infty$.

(iii) if $\tau = 0$, then $\|w_k - w_*\|^2 \leq (1 - \alpha\mu)^k \|w_0 - w_*\|^2 + 2\epsilon R/\mu^2 + \alpha^2 \gamma_{\text{max}}^2 C^2/\mu$,

where $C$ is an upper-bound on the norm of the component function gradients, i.e., $\max_{i \in V} \sup_{w \in W} \|\nabla f_i(w)\| \leq C$, and $\gamma_{\text{max}} = \max_{j \in S} \gamma_j$ is the largest per-element step size, and $R = \min\{d_0, (r\gamma_{\text{max}} C + \epsilon)/\mu\}$, where $d_0 = \|w_0 - w_*\|$ is the initial distance to the optimum $w_*$.

All the proofs can be found in the Appendix. The above theorem shows that IG on $S$ converges at the same rate $O(1/\sqrt{k})$ of IG on the entire data set $V$. However, compared to IG on $V$, the $|V|/|S|$ speedup of IG on $S$ comes at the price of getting an extra error term, $2\epsilon R/\mu^2$.

### 4.4. Convergence for Smooth and Strongly Convex $f$

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

**Theorem 2.** Assume that $f$ is strongly convex and let $f_i(w), i = 1, 2, \ldots, n$ be convex and twice continuously differentiable component functions with Lipschitz gradients on $W$. Supposed that $S$ is a weighted subset of size $r$ obtained by CRAIG that estimates the full gradient by an error of at most $\epsilon$, i.e.,

$$\max_{w \in W} \|\sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \nabla f_j(w)\| \leq \epsilon.$$

Then for the iterates $\{w_k = w_0^k\}$ generated by applying IG to $S$ with per-epoch stepsize $\alpha_k = \alpha/k^r$ with $\alpha > 0$ and $\tau \in [0, 1]$, we have

(i) if $\tau = 1$, then $\|w_k - w_*\| \leq 2\epsilon/\mu + \alpha\beta C\gamma_{\text{max}}^2/k\mu$,

(ii) if $0 < \tau < 1$, then $\|w_k - w_*\| \leq 2\epsilon/\mu$, for $k \to \infty$. 
Appendix B.1 only hold for points with similar labels. Thus, theoretically we need to select subsets separately. For neural networks, we observed that separately selecting subsets from each class helps the performance. We separately tune each method so that it performs at its best.

5.1. Convex Experiments
In our convex experiments, we apply CRAIG to SGD, as well as SVRG (Johnson & Zhang, 2013), and SAGA (Defazio et al., 2014). We apply L2-regularized logistic regression: \( f_i(x) = \ln(1 + \exp(-w^T x_i y_i)) + 0.5\lambda w^T w \) to classify the following two datasets from LIBSVM: (1) \texttt{covtype.binary} including 581,012 data points of 54 dimensions, and (2) \texttt{ljcnn1} including 49,990 training and 91,701 test data points of 22 dimensions. As \texttt{covtype} does not come with labeled test data, we randomly split the training data into halves to make the training/test split (training and set sets are consistent for different methods).

For the convex experiments, we tuned the learning rate for each method (including the random baseline) by preferring smaller training loss from a large number of parameter combinations for two types of learning scheduling: exponential decay \( \alpha_k = \alpha_0 b^k \) and \( k \)-inverse \( \alpha_k = \alpha_0 (1 + bk)^{-1} \) with parameters \( \alpha_0 \) and \( b \) to adjust. For convergence of IG to \( 2\epsilon/\mu \) neighborhood of the optimal solution, we require that \( \sum_{k=0}^{\infty} \alpha_k = \infty \), and \( \sum_{k=0}^{\infty} \alpha_k^2 = 0 \) (Nedić & Bertsekas, 2001). Hence, while the convergence of exponentially decaying learning rate is not theoretically guaranteed, it often worked better in our experiments. Furthermore, following (Johnson & Zhang, 2013) we set \( \lambda \) to \( 10^{-5} \).

CRAIG effectively minimizes the loss. Figure 1(top) compares training loss residual of SGD, SVRG, and SAGA on the 10% CRAIG set (blue), 10% random set (green), 10% selected subset (blue) vs. 10% random subset (green) vs. entire data set (orange).
and the full dataset (orange). CRAIG effectively minimizes the training data loss (blue line) and achieves the same minimum as the entire dataset training (orange set) but much faster. Also notice that training on the random 10% subset of the data does not effectively minimize the training loss.

CRAIG has a good generalization performance. Figure 1(bottom) shows the test error rate of models trained on CRAIG vs. random vs. the full data. Notice that training on CRAIG subsets achieves the same generalization performance (test error rate) as training on the full data.

CRAIG achieves significant speedup. Figure 1 also shows that CRAIG achieves a similar training loss (top) and test error rate (bottom) as training on the entire set, but much faster. In particular, we obtain a speedup of 2.75x, 4.5x, 2.5x from applying IG, SVRG and SAGA on the subsets of size 10%, 20%, 50% from covtype obtained by CRAIG. Furthermore, Figure 3 compares the speedup achieved by CRAIG to reach a similar loss residual as that of SGD for subsets of size 10%, 20%, 50%, 90% of Ijcnn1. We get a 5.6x speedup by applying SGD to subsets of size 30% obtained by CRAIG.

CRAIG gradient approximation is accurate. Figure 2 demonstrates the norm of the difference between the weighted gradient of the subset found by CRAIG and the full gradient compared to the theoretical upper-bound $\epsilon$ specified in Eq. (8). The gradient difference is calculated by sampling the full gradient at various points in the parameter space. Gradient differences are then normalized by the largest norm of the sampled full gradients. The figure also compares the normed gradient difference between gradients of several random subsets $S$ where each data point is weighted by $|V|/|S|$. Notice that CRAIG’s gradient estimate is more accurate than the gradient estimate obtained by the same-size random subset of points (which is how methods like SGD approximate the gradient). This demonstrates that our gradient approximation in Eq. (8) is reliable in practice.

5.2. Non-convex Experiments

Our non-convex experiments involve applying CRAIG 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 $\lambda = 10^{-4}$ 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-20 for CIFAR10 with convolution, average pooling and dense layers with softmax outputs and L2 regularization with $\lambda = 10^{-4}$. 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 data sets are normalized into $[0, 1]$ by division with 255. In all these experiments, we report average test accuracy across 10 trials.

CRAIG achieves considerable speedup. Figure 4 shows training loss and test accuracy for training a 2-layer neural net on MNIST. For this problem, we used a constant learning rate of $10^{-2}$. Here, we apply CRAIG to select a subset of 50% of the data at the beginning of every epoch and train only on the selected subset with the corresponding per-element stepsize. Interestingly, in addition to achieving a speedup of 2x to 3x for training the network, the subsets selected by CRAIG provide a better generalization performance compared to models trained on the entire dataset.
CRAIG is data-efficient for training neural networks.

Figure 5 shows test accuracy vs. the fraction of data points selected for training ResNet-20 on CIFAR10. We trained the network for 200 epochs, and used the standard learning rate schedule for training ResNet-20 on CIFAR10. i.e., we start with initial learning rate of 0.1, and exponentially decay the learning rate by a factor of 0.1 at epochs 100 and 150. To prevent weights from diverging when training with subsets of all sizes, we used linear learning rate warm-up for 20 epochs from 0. For optimization we used SGD with a momentum of 0.9.

Figure 5a shows the test accuracy when at the beginning of every epoch a subset of size 1%, 2%, 3%, 4%, 5%, 10%, or 20% is selected at random or by CRAIG. SGD is then applied to training on the selected subsets. The x-axis shows the fraction of training data points that are used by SGD during the training process. Note that for a given subset size, backpropagation is done on the same number of data points for CRAIG and random. However, CRAIG selects a smaller number of distinct data points during the training. Therefore, CRAIG is data-efficient for training neural networks.

Insights from CRAIG subsets. Figure 6 shows a subset of images selected by CRAIG from CIFAR10. Subsets are selected at the (a) beginning of training (epoch 1), (b) middle of training (epoch 100), and (c) end of training (epoch 200). We notice that during the training, semantic redundancies decrease considerably, and coreset images better represent various types of images (that are more difficult to learn) in every class.

6. Conclusion

We developed a method, CRAIG, for selecting a subset (coreset) of data points with their corresponding per-element stepsizes to speed up iterative gradient (IG) methods. 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, the subset can be found using a fast greedy algorithm. We proved that IG on subsets $S$ found by CRAIG converges at the same rate as IG on the entire dataset $V$, while providing a $|V|/|S|$ speedup. In our experiments, we showed that various IG methods, including SGD, SAGA, and SVRG run up to 6x faster on convex and up to 3x on non-convex problems on subsets found by CRAIG while achieving practically the same training loss and test error.
Acknowledgement

This work was supported in part by the SNSF P2EZP2_172187, and the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. We also gratefully acknowledge the support of DARPA under Nos. FA865018C7880 (ASED), N66001192403 (MCS); ARO under Nos. W911NF-16-1-0342 (MURI), W911NF-16-1-0171 (DURIP); NSF under Nos. OAC-1835598 (CINES), OAC-1934578 (HDR), CCF-1918940 (Experiments), IIS-2030477 (RAPID); Stanford Data Science Initiative, Wu Tsai Neurosciences Institute, Chan Zuckerberg Biohub, Amazon, Boeing, Chase, Docomo, Hitachi, Huawei, JD.com, NVIDIA, Dell. J. L. is a Chan Zuckerberg Biohub investigator.

References


