Robust Submodular Partitioning and its Applications to Distributed Machine Learning

Kai Wei[1] **Rishabh Iyer**[1] **Shengjie Wang**[2] **Wenruo Bai**[1] **Jeff Bilmes**[1]

[1] Department of Electrical Engineering, University of Washington
[2] Department of Computer Science, University of Washington
{kaiwei, rkiyer, wangsj, wrbai, bilmes}@u.washington.edu

Abstract

We investigate a robust submodular data partitioning problem called *max-min submodular fair allocation* (SFA) [7]. While SFA has been studied in the theory community [6, 7, 9], existing work has focused on tight approximation guarantees, and the resultant algorithms are not generally scalable to large real-world applications. In the present paper, we bridge this gap, by proposing two variants of a greedy algorithm that not only scale to large datasets but that also achieve theoretical approximation guarantees comparable to the state-of-the-art. We show that SFA can be applied in distributed training of statistical models. Lastly we empirically demonstrate the efficacy of our algorithms to obtain data partitioning for distributed optimization of convex and deep neural network objectives.

1 Introduction

This paper studies a robust partitioning problem of the following form:

$$\text{Problem 1:} \qquad \max_{\pi \in \Pi} \min_i f_i(A_i^\pi) \tag{1}$$

where the set of sets $\pi = (A_1^\pi, A_2^\pi, \cdots, A_m^\pi)$ is a partition of a finite set V (i.e, $\cup_i A_i^\pi = V$ and $\forall i \neq j, A_i^\pi \cap A_j^\pi = \emptyset$), and Π refers to the set of all partitions of V into m blocks. In general, Problem 1 is hopelessly intractable, even to approximate, but we assume that the $f_1, f_2, \cdots, f_m$ are all monotone non-decreasing (i.e., $f_i(S) \leq f_i(T)$ whenever $S \subseteq T$), normalized ($f_i(\emptyset) = 0$), and submodular [5] (i.e., $\forall S, T \subseteq V$, $f_i(S) + f_i(T) \geq f_i(S \cup T) + f_i(S \cap T)$). These assumptions allow us to develop fast, simple, and scalable algorithms that have approximation guarantees, as is done in this paper. These assumptions, moreover, allow us to retain the naturalness and applicability of Problems 1 to a wide variety of practical problems. Submodularity is a natural property in many real-world ML applications [22, 24, 13, 21, 11, 8, 3, 10, 19]. When minimizing, submodularity naturally models notions of interacting costs and complexity, while when maximizing it readily models notions of diversity, summarization quality, and information. Hence, Problem 1 asks for a partition whose blocks each (and that collectively) are a good, say, summary of the whole. We refer to Problem 1 as *Robust Submodular Partitioning*. We further categorize this problem depending on if the f_i's are identical to each other (*homogeneous*) or not (*heterogeneous*).[1] The heterogeneous case clearly generalizes the homogeneous setting, but as we will see, the additional homogeneous structure can be exploited to provide more efficient and/or tighter algorithms.

Previous work: SFA has been studied mostly in the heterogeneous setting. When f_i's are all modular, the tightest algorithm, so far, is to iteratively round an LP solution achieving $O(1/(\sqrt{m}\log^3 m))$ approximation [1], whereas the problem is NP-hard to $1/2 + \epsilon$ approximate for any $\epsilon > 0$ [7]. When

[1]Similar sub-categorizations have been called the "uniform" vs. the "non-uniform" case in the past [16, 6].

f_i's are submodular, [7] gives a matching-based algorithm with a factor $1/(n-m+1)$ approximation that performs poorly when $m \ll n$. [9] proposes a binary search algorithm yielding an improved factor of $1/(2m-1)$. Another approach approximates each submodular function by its ellipsoid approximation (non-scalable) and reduces SFA to its modular version leading to an approximation factor of $O(\sqrt{n}m^{1/4}\log n \log^{3/2} m)$. These approaches are theoretically interesting, but they either do not fully exploit the structure of the problem or do not scale to large problems.

Applications: An important machine learning application of SFA, evaluated in Section 3, is distributed training of statistical models. As data set sizes grow, the need for statistical training procedures tolerant of the distributed data partitioning becomes more important. Existing schemes are often developed and performed assuming data samples are distributed to their computational clients in an arbitrary or random fashion. As an alternate strategy, if the data is intelligently partitioned such that each block of samples can itself lead to a good approximate solution, a consensus amongst the distributed results could be reached more quickly than when under a poor partitioning. Submodular functions can in fact express the value of a subset of training data for certain machine learning risk functions, e.g., [19] in the case of classification. Using these functions within Problem 1, one can expect a partitioning (by formulating the problem as an instance of Problem 1, where each block is a good *representative* of the entire set, thereby achieving faster convergence in distributed settings. We demonstrate empirically, in Section 3, that this provides better results on several machine learning tasks, including the training of deep neural networks.

Our Contributions: We first give a simple and scalable greedy algorithm (GreedMax), and show a factor of $1/m$ in the homogeneous setting, improving the state-of-the-art factor of $1/(2m-1)$ under the heterogeneous setting [9]. For the heterogeneous setting, we propose a "saturate" greedy algorithm (GreedSat) that iteratively solves instances of submodular welfare problems [18]. We show GreedSat has a bi-criterion guarantee of $(1/2-\delta, \delta/(1/2+\delta))$, which ensures at least $\lceil m(1/2-\delta)\rceil$ blocks receive utility at least $\delta/(1/2+\delta)OPT$ for any $0 < \delta < 1/2$. Lastly, we demonstrate the efficacy of SFA to distributed machine learning, including ADMM and neural network training.

2 Approximation Algorithms for SFA

Notation: we define $f(j|S) \triangleq f(S \cup j) - f(S)$ as the gain of $j \in V$ in the context of $S \subseteq V$. We assume w.l.o.g. that the ground set is $V = \{1, 2, \cdots, n\}$.

We approach SFA from the perspective of the greedy algorithms. In particular we introduce two variants of a greedy algorithm – GreedMax (Alg. 1) and GreedSat (Alg. 2), suited for the homogeneous and heterogeneous settings, respectively.

GreedMax: The key idea of GreedMax (see Alg. 1) is to greedily add an item with the maximum marginal gain to the block whose current solution is minimum. Initializing $\{A_i\}_{i=1}^m$ with the empty sets, the greedy flavor also comes from that it incrementally grows the solution by greedily improving the overall objective $\min_{i=1,\dots,m} f_i(A_i)$ until $\{A_i\}_{i=1}^m$ forms a partition. Besides its simplicity, Theorem 2.1 offers the optimality guarantee.

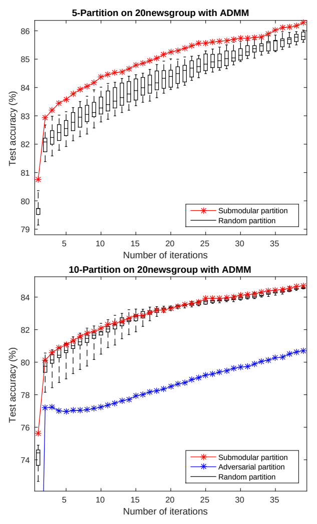
Algorithm 1: GreedMax
```
1: Input: $f, m, V$.
2: Let $A_1 =, \dots, = A_m = \emptyset$; $R = V$.
3: while $R \neq \emptyset$ do
4:     $j^* \in \text{argmin}_j f(A_j)$;
5:     $a^* \in \text{argmax}_{a \in R} f(a|A_{j^*})$
6:     $A_{j^*} \leftarrow A_{j^*} \cup \{a^*\}$; $R \leftarrow R \setminus a^*$
7: end while
8: Output $\{A_i\}_{i=1}^m$.
```

Theorem 2.1. GreedMax *achieves a guarantee of* $1/m$ *under the homogeneous setting.*

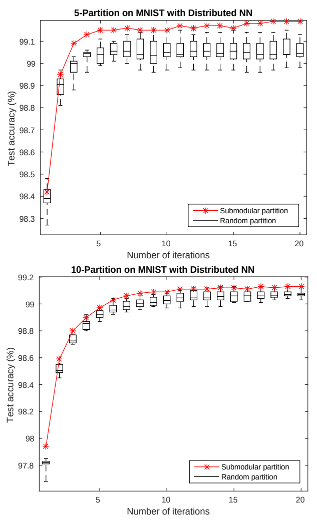
All proofs for this paper are deferred to [20]. By assuming the homogeneity of the f_i's, we obtain a very simple $1/m$-approximation algorithm improving upon the state-of-the-art factor $1/(2m-1)$ [9]. Thanks to the lazy evaluation trick as described in [12], Line 5 in Alg. 1 need not to recompute the marginal gain for every item in each round, leading GreedMax to scale to large data sets.

Algorithm 2: GreedSat
```
1: Input: $\{f_i\}_{i=1}^m, m, V, \alpha$.
2: Let $\bar{F}^c(\pi) = \frac{1}{m}\sum_{i=1}^m \min\{f_i(A_i^\pi), c\}$.
3: Let $c_{\min} = 0$, $c_{\max} = \min_i f_i(V)$
4: while $c_{\max} - c_{\min} \geq \epsilon$ do
5:     $c = \frac{1}{2}(c_{\max} + c_{\min})$
6:     $\hat{\pi}^c \in \text{argmax}_{\pi \in \Pi} \bar{F}^c(\pi)$
7:     if $\bar{F}^c(\hat{\pi}^c) < \alpha c$ then
8:         $c_{\max} = c$
9:     else
10:        $c_{\min} = c$; $\hat{\pi} \leftarrow \hat{\pi}^c$
11:    end if
12: end while
13: Output: $\hat{\pi}$.
```

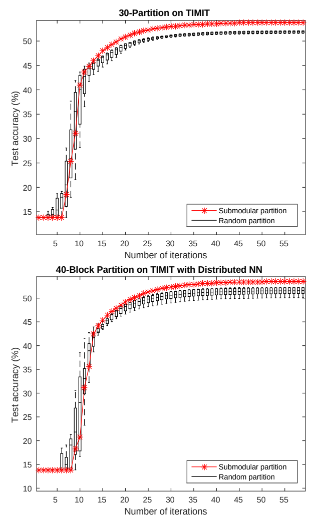
GreedSat: Though simple and working well for the homogeneous setting, GreedMax performs arbitrarily poorly under the heterogeneous setting. To this end we provide another algorithm – "Saturate" Greedy (GreedSat,



(a)20Newsgroups



(b) MNIST



(c) TIMIT

Figure 1: Comparison between submodular and random partitions for distributed ML, including ADMM (Fig 1a) and distributed deep neural nets (Fig 1b) and (Fig 1c). For the box plots, the central mark is the median, the box edges are 25th and 75th percentiles, and the bars indicate the best and worst cases. In the right two columns, the adversarial partitions are so bad that they are off the plots.

see Alg. 2). The key idea of GREEDSAT is to relax SFA to a much simpler problem – Submodular Welfare (SWP) [18]. Similar in flavor to the one proposed in [10] GREEDSAT defines an intermediate objective $\bar{F}^c(\pi) = \sum_{i=1}^m f_i^c(A_i^\pi)$, where $f_i^c(A) = \frac{1}{m}\min\{f_i(A), c\}$ (Line 2). The parameter c controls the saturation in each block. f_i^c satisfies submodularity for each i. Unlike SFA, the combinatorial optimization problem $\max_{\pi\in\Pi} \bar{F}^c(\pi)$ (Line 6) is much easier and is an instance of SWP. In this work, we solve Line 6 by the efficient greedy algorithm as described in [4] with a factor $1/2$. It should be clear that one can also implement a computationally expensive multi-linear relaxation algorithm as given in [18] to solve Line 6 with a tight factor $(1 - 1/e)$. Setting the input argument $\alpha = 1/2$ as the approximation factor for Line 6, the essential idea of GREEDSAT is to perform a binary search over the parameter c to find the largest c^* such that the returned solution $\hat{\pi}^{c^*}$ for the instance of SWP satisfies $\bar{F}^{c^*}(\hat{\pi}^{c^*}) \geq \alpha c^*$. GREEDSAT terminates in solving $O(\log(\frac{\min_i f_i(V)}{\epsilon}))$ instances of SWP. Theorem 2.2 gives a bi-criterion optimality guarantee.

Theorem 2.2. *Given $\epsilon > 0$, $0 \leq \alpha \leq 1$ and any $0 < \delta < \alpha$, GREEDSAT finds a partition such that at least $\lceil m(\alpha - \delta)\rceil$ blocks receive utility at least $\frac{\delta}{1-\alpha+\delta}(\max_{\pi\in\Pi}\min_i f_i(A_i^\pi) - \epsilon)$.*

For any $0 < \delta < \alpha$ Theorem 2.2 ensures that the top $\lceil m(\alpha - \delta)\rceil$ valued blocks in the partition returned by GREEDSAT are $(\delta/(1-\alpha+\delta) - \epsilon)$-optimal. δ controls the trade-off between the number of top valued blocks to bound and the performance guarantee attained for these blocks. The smaller δ is, the more top blocks are bounded, but with a weaker guarantee. We set the input argument $\alpha = 1/2$ as the worst-case performance guarantee for solving SWP so that the above theoretical analysis follows. However, the worst-case factor is often achieved by very contrived examples of submodular functions. For the ones used in practice, the greedy algorithm often leads to near-optimal solution [10]. Setting α as the actual performance guarantee for SWP (often very close to 1), the bound can be significantly improved. In practice, we suggest setting $\alpha = 1$.

3 Experiments and Conclusions

In this section empirically evaluate the algorithms proposed for SFA on real-world data partitioning applications including distributed ADMM and distributed deep neural network training.

Distributed Convex Optimization: We first consider data partitioning for distributed convex optimization. We evaluate on a text categorization task. We use 20 Newsgroup data set, which consists of 18,774 articles divided almost evenly across 20 classes. The text categorization task is to classify an article into one newsgroup (of twenty) to which it was posted. We randomly split

2/3 and 1/3 of the whole data as the training and test data. The task is solved as a multi-class classification problem, which we formulate as an L-2 regularized logistic regression. We solve this convex optimization problem in a distributive fashion, where the data samples are partitioned and distributed across multiple machines. In particular we implement an ADMM algorithm as described in [2] to solve the distributed convex optimization problem.

We formulate the data partitioning problem as an instance of SFA under the homogeneous setting. In the experiment, we solve the data partitioning using GREEDMAX, since it is efficient and attains the tightest guarantee among all algorithms proposed for this setting. Note, however, GREEDSAT may also be used in the experiment. We model the utility of a data subset using the feature-based submodular function [21, 19, 17], which has the form: $f_{\text{fea}}(A) = \sum_{u \in \mathcal{U}} m_u(V) \log m_u(A)$, where $\mathcal{U}$ is the set of "features", $m_u(A) = \sum_{a \in A} m_u(a)$ with $m_u(a)$ measuring the degree that the article a possesses the feature $u \in \mathcal{U}$. In the experiments, we define $\mathcal{U}$ as the set of all words occurred in the entire data set and $m_u(a)$ as the number of occurrences of the word $u \in \mathcal{U}$ in the article a. f_{fea} is in the form of a sum of concave over modular functions, hence is monotone submodular [15]. The class of feature-based submodular function has been widely applied to model the utility of a data subset on a number of tasks, including speech data subset selection [21, 23], and image summarization [17]. Moreover f_{fea} has been shown in [19] to model the log-likelihood of a data subset for a Naïve Bayes classifier.

We compare the submodular partitioning with the random partitioning for $m = 5$ and $m = 10$. We test with 10 instances of random partitioning. The results are plotted in Fig 1a. For $m = 10$ we also run an instance on an adversarial partitioning, where each block is formed by grouping every two of the 20 classes in the training data. We observe submodular partitioning converges faster than the random partitioning, both of which perform significantly better than the adversarial partition. In particular significant and consistent improvement over the best of 10 random instances is achieved by the submodular partition across all iterations when $m = 5$.

Distributed Neural Network: Next we evaluate our framework on the distributed neural network training. We test on two tasks: 1) handwritten digit recognition on the MNIST database; 2) phone classification on the TIMIT data. The data for the handwritten digit recognition task consists of 60,000 training and 10,000 test samples. Each data sample is an image of handwritten digit. The training and test data are almost evenly divided into 10 different classes. For the phone classification task, the data consists of 1,124,823 training and 112,487 test samples. Each sample is a frame of speech. The training data is divided into 50 classes, each of which corresponds to a phoneme. The goal of this task to classify each speech sample into one of the 50 phone classes.

A 4-layer DNN model is applied for the MNIST experiments, and we train a 5-layered NN for the TIMIT experiments. We apply the same distributed training procedure for both tasks. Given a partitioning of the training data, we distributively solve m instances of sub-problems in each iteration. We define each sub-problem on a separate block of the data. We employ the stochastic gradient descent as the solver on each instance of the sub-problem. In the first iteration we use a randomly generated model as the initial model shared among the m sub-problems. Each sub-problem is solved with 10 epochs of the stochastic gradient decent training. We then average the weights in the m resultant models to obtain a consensus model, which is used as the initial model for each sub-problem in the successive iteration. Note that this distributed training scheme is similar to the ones presented in [14].

The submodular partitioning for both tasks is obtained by solving the homogeneous case of SFA using GREEDMAX on a form of clustered facility location, as proposed and used in [19]. The function is defined as follows: $f_{\text{c-fac}}(A) = \sum_{y \in \mathcal{Y}} \sum_{v \in V^y} \max_{a \in A \cap V^y} s_{v,a}$, where $s_{v,a}$ is the similarity measure between sample v and a, $\mathcal{Y}$ is the set of class labels, and V^y is the set of samples in V with label $y \in \mathcal{Y}$. Note $\{V^y\}_{y \in \mathcal{Y}}$ forms a disjoint partitioning of the ground set V. In both the MNIST and TIMIT experiments we compute the similarity $s_{v,a}$ as the RBF kernel between the feature representation of v and a. [19] show that $f_{\text{c-fac}}$ models the log-likelihood of a data subset for a Nearest Neighbor classifier. They also empirically demonstrate the efficacy of $f_{\text{c-fac}}$ in the case of neural network based classifiers.

We also run 10 instances of random partitioning as the baseline. As shown in Figure 1b and 1c, the submodular partitioning outperforms the random baseline. The adversarial partitioning, which is formed by grouping items with the same class, cannot even be trained in both cases.

As can be seen, in all cases submodular partitioning is scalable and yields improved results.

References

[1] A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *SICOMP*, 2010.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 2011.

[3] A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv preprint arXiv:1102.3975*, 2011.

[4] M. Fisher, G. Nemhauser, and L. Wolsey. An analysis of approximations for maximizing submodular set functions—II. In *Polyhedral combinatorics*, 1978.

[5] S. Fujishige. *Submodular functions and optimization*, volume 58. Elsevier, 2005.

[6] M. Goemans, N. Harvey, S. Iwata, and V. Mirrokni. Approximating submodular functions everywhere. In *SODA*, 2009.

[7] D. Golovin. Max-min fair allocation of indivisible goods. *Technical Report CMU-CS-05-144*, 2005.

[8] S. Jegelka and J. Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In *CVPR*, 2011.

[9] S. Khot and A. Ponnuswami. Approximation algorithms for the max-min allocation problem. In *APPROX*, 2007.

[10] A. Krause, B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. In *JMLR*, 2008.

[11] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. In *JMLR*, 2008.

[12] M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, 1978.

[13] K. Nagano, Y. Kawahara, and S. Iwata. Minimum average cost clustering. In *Advances in Neural Information Processing Systems*, pages 1759–1767, 2010.

[14] D. Povey, X. Zhang, and S. Khudanpur. Parallel training of deep neural networks with natural gradient and parameter averaging. *arXiv preprint arXiv:1410.7455*, 2014.

[15] P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *NIPS*, 2010.

[16] Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *FOCS*, 2008.

[17] S. Tschiatschek, R. K. Iyer, H. Wei, and J. A. Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Advances in Neural Information Processing Systems*, pages 1413–1421, 2014.

[18] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, 2008.

[19] K. Wei, R. Iyer, and J. Bilmes. Submodularity in data subset selection and active learning. In *ICML*, 2015.

[20] K. Wei, R. Iyer, S. Wang, W. Bai, and J. Bilmes. Mixed robust/average submodular partitioning: Fast algorithms, guarantees, and applications: NIPS 2015 Extended Supplementary.

[21] K. Wei, Y. Liu, K. Kirchhoff, C. Bartels, and J. Bilmes. Submodular subset selection for large-scale speech training data. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Florence, Italy, 2014.

[22] K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes. Using document summarization techniques for speech data subset selection. In *North American Chapter of the Association for Computational Linguistics/Human Language Technology Conference (NAACL/HLT-2013)*, Atlanta, GA, June 2013.

[23] K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes. Unsupervised submodular subset selection for speech data. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Florence, Italy, 2014.

[24] J. Zheng, Z. Jiang, R. Chellappa, and J. P. Phillips. Submodular attribute selection for action recognition in video. In *NIPS*, 2014.