Algorithms for Optimizing the Ratio of Submodular Functions

Wenruo Bai WRBAI@UW.EDU
Rishabh Iyer RKIYER@U.WASHINGTON.EDU
Kai Wei WEIKAI.HUST@GMAIL.COM
Jeff Bilmes BILMES@UW.EDU
University of Washington, Seattle, WA 98195, USA

Abstract

We investigate a new optimization problem involving minimizing the Ratio of two Submodular (RS) functions. We argue that this problem occurs naturally in several real world applications. We then show the connection between this problem and several related problems including minimizing the difference between submodular functions (Iyer & Bilmes, 2012b; Narasimhan & Bilmes, 2005), and to submodular optimization subject to submodular constraints (Iyer & Bilmes, 2013). We show that RS optimization can be solved with bounded approximation factors. We also provide a hardness bound and show that our tightest algorithm matches the lower bound up to a log factor. Finally, we empirically demonstrate the performance and good scalability properties of our algorithms.

1. Introduction

A set function $f : 2^V \to \mathbb{R}_+$ is said to be *submodular* (Fujishige, 2005) if for all subsets $S, T \subseteq V$, it holds that $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$. Defining $f(j|S) \triangleq f(S \cup j) - f(S)$ as the gain of $j \in V$ in the context of $S \subseteq V$, then f is submodular if and only if $f(j|S) \geq f(j|T)$ for all $S \subseteq T$ and $j \notin T$. The function f is monotone iff $f(j|S) \geq 0, \forall j \notin S, S \subseteq V$. W.l.o.g., we assume the ground set is $V = \{1, 2, \cdots, n\}$. While general set function optimization is often intractable, many forms of submodular function optimization can be solved near optimally or even optimally in certain cases, and hence submodularity is also often called the discrete analog of convexity (Lovász, 1983). Submodularity, moreover, is inherent in a large set of real-world machine learning applications, therefore making them useful in practice.

In this paper, we study a new class of discrete optimization problems that have the following form:

$$\text{Problem 1: } \min_{\emptyset \subset X \subseteq V} \frac{f(X)}{g(X)}, \qquad (1)$$

where f and g are monotone non-decreasing submodular functions. We call the objective $\frac{f(A)}{g(A)}$ a ratio of submodular (RS) function. We do not require that either f or g is normalized, we only assume $f(\emptyset) \geq 0$ and $g(\emptyset) \geq 0$. We assume, w.l.o.g., that both f and g satisfy $f(v) > 0, g(v) > 0, \forall v \in V$, since we may simply remove any item $v \in V$ for consideration if $g(v) = 0$ and add any item $v \in V$ to the solution if $f(v) = 0$. As a consequence of this assumption and the monotonicity of f and g, we have that $f(A) > 0, g(A) > 0, \forall \emptyset \subset A \subseteq V$.

We call this problem RS minimization. In Section 5, we extend the algorithms here to handle non-monotone submodular functions. We also consider a related problem called RS maximization:

$$\text{Problem 2: } \max_{\emptyset \subset X \subseteq V} \frac{g(X)}{f(X)}. \qquad (2)$$

We observe that RS Minimization and RS Maximization are equivalent, in that an algorithm for Problem 1 also solves Problem 2. To be precise, given an α-approximation algorithm ($\alpha > 1$) for Problem 1, one can achieve a $1/\alpha$ approximation for Problem 2, using the solution obtained by the algorithm for Problem 1.

1.1. Applications

In the below, we describe how Problems 1 and 2 appear naturally in several machine learning applications.

Maximizing the F-Measure in Information Retrieval: Consider the problem where one is given a set U of objects (e.g., documents, images etc.) which can be expressed as a bag of words W. One can construct a bipartite graph $G(U, W, E)$, where U is the set of objects, W is the set of words, and the edge $e_{u,w} \in E$ between the object u and the word w exists if the word w is present in the object u. Define a neighborhood function $\Gamma : 2^U \to 2^W$ on the bipartite

Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

graph G that maps from any subset of the objects $X \subseteq U$ to the set of words $\Gamma(X) \subseteq W$ contained in the objects. We are interested in the information retrieval problem of finding a set of objects that cover exactly a set of target words $T \subseteq W$. The quality of any subset $X \subseteq U$ may be measured as the F-Measure of the coverage on the target words T. More formally, the quality measure can be defined as:

$$F(X) = \frac{2|\Gamma(X) \cap T|}{|T| + |\Gamma(X)|}. \tag{3}$$

The goal is to find a set of objects X that maximizes the F-measure $F(X)$. Note that both $|\Gamma(X) \cap T|$ and $|T| + |\Gamma(X)|$ are monotone submodular. Hence the problem is an instance of RS maximization (Problem 2).

Normalized Cuts and Ratio Cuts: Another application of RS optimization is the normalized cut and ratio cut problem, which have been extensively used in image segmentation and clustering (Shi & Malik, 2000; Kawahara et al., 2011). Let $G = (V, E)$ be a similarity graph defined on the set of vertices V, where $w_{a,a'}$ is the edge weight between the vertex a and $a' \in V$ and measures the similarity between these two vertices. Let $h(A) = \sum_{a \in A} \sum_{a' \in V \setminus A} w_{a,a'}$ be the graph cut function defined on the graph G, and let $m(A) = \sum_{a \in A} \sum_{v \in V} w_{a,v}$ be the function that measures the association of the subset A to the ground set V. The normalized cut problem as defined in (Shi & Malik, 2000) is to minimize the following objective:

$$\frac{h(A)}{m(A)} + \frac{h(A)}{m(V) - m(A)}, \tag{4}$$

which can further simplified as follows:

$$\frac{h(A)m(V)}{m(A)(m(V) - m(A))}. \tag{5}$$

Note that $m(V)$ is a constant, and both $h(A)$ and $m(A)(m(V) - m(A))$ are symmetric submodular functions. Therefore, the normalized cuts problem can be formulated as a non-monotone instance of RS minimization (Problem 1). A similar case is given in (Narasimhan & Bilmes, 2007).

Maximizing Diversity & Minimizing Cooperative Costs: A final set of applications are related to simultaneously maximizing diversity or coverage, while minimizing cooperative costs. Applications of this involve sensor placement, feature selection (Krause et al., 2008; Iyer & Bilmes, 2012b; Liu et al., 2013), and data subset selection (Lin & Bilmes, 2011; Wei et al., 2013; 2015a). While these problems are often modeled as a difference of submodular functions, or constrained submodular optimization, one can also model them as a ratio of submodular functions, where the submodular function f models the cooperative costs while g models information and utility. The ratio $\frac{g(A)}{f(A)}$ naturally models the cost normalized utility of the set A. Maximizing $\frac{g(A)}{f(A)}$ (Problem 2) leads to the best cost normalized subset A.

Algorithm 1 A $(1+\epsilon)$-approximation algorithm for RS minimization using an exact algorithm for DS minimization

```
1: Input: $f$, $g$, $\epsilon \in [0, 1)$ and an exact algorithm for DS minimization.
2: Output: A $(1+\epsilon)$-approximate solution for Prob. 1
3: Set $\lambda_{\max} \leftarrow \frac{f(A)}{g(A)}$ for arbitrary $A \subseteq V$, and $\lambda_{\min} \leftarrow 0$.
4: while $\lambda_{\max} > (1+\epsilon)\lambda_{\min}$ do
5:   $\bar{\lambda} \leftarrow \frac{\lambda_{\min}+\lambda_{\max}}{2}$.
6:   $\hat{X} \leftarrow \text{argmin}_{X \supset \emptyset}[f(X) - \bar{\lambda} g(X)]$.
7:   if $\frac{f(\hat{X})}{g(\hat{X})} \geq \bar{\lambda}$ then
8:     $\lambda_{\min} \leftarrow \bar{\lambda}$
9:   else
10:    $\lambda_{\max} \leftarrow \bar{\lambda}$
11:  end if
12: end while
13: Return $\hat{X} \leftarrow \text{argmin}_X [f(X) - \bar{\lambda} g(X)]$.
```

1.2. RoadMap of this Paper

The rest of the paper is organized as follows. We first describe connections between RS minimization and related problems studied in the literature (Section 2). In particular, we show how this is closely related to the problem of minimizing the difference between submodular functions and to the problem of submodular optimization subject to submodular lower bound and upper bound constraints. In Section 3, we provide several approximation algorithms along with the analysis of their approximation guarantees for RS minimization. The algorithms include GreedyRatio, Binary Search, Majorization-Minimization, and Ellipsoidal Approximations. In Section 4, we prove matching hardness bounds for this Problem. In Section 5, we consider extensions of RS minimization where f and g may be supermodular and/or non-monotone submodular. Empirical evaluations on synthetic data are given in Section 6.

2. Connections to Related Problems

Connections to DS optimization: A problem related to the RS minimization is the Difference of Submodular (DS) minimization defined as follows:

$$\text{Problem: } \min_{X \subseteq V} [f(X) - \lambda g(X)], \tag{6}$$

where $\lambda \geq 0$. We call the objective $f(X) - \lambda g(X)$ a difference of submodular (DS) function. We show below that, in fact, an exact algorithm for DS minimization can be used as a subroutine to also solve RS minimization via a simple binary search scheme as described in Alg. 1.

Lemma 2.1. *Given $\epsilon > 0$ and an exact algorithm for solving DS minimization (Problem 6), Algorithm 1 provides a $(1+\epsilon)$-approximation for RS minimization (Problem 1), by solving $O(\log(1/\epsilon))$ instances of DS minimization.*

Proof. When the algorithm terminates, the following holds: $\min_X[f(X) - \lambda_{\min} g(X)] \geq 0$, which implies that $\frac{f(X)}{g(X)} \geq \lambda_{\min}, \forall X \subset V$. Therefore, it holds that: $\frac{f(\hat{X})}{g(\hat{X})} \leq \lambda_{\max} \leq (1+\epsilon)\lambda_{\min} \leq (1+\epsilon)\min_X \frac{f(X)}{g(X)}$. □

While RS minimization and DS minimization are closely related, we show below the class of set functions representable as an RS function is strictly contained by the class of DS functions. Thus, the DS minimization problem encapsulates a strictly larger class of combinatorial optimization problems.

Lemma 2.2. *Any RS function can be expressed as a DS function. However, there exists an instance of a DS function that cannot be represented as an RS function.*

Proof. The first half of the lemma holds since any RS function is a set function, and any set function can be expressed as a DS function (Narasimhan & Bilmes, 2005).

To show the second half of the Lemma, we give a counter-example as follows: Let $V = \{1, 2\}$, $h(X) = f_1(X) - g_1(X)$ where $f_1(X) = \sqrt{2|X|}$ and $g_1(X) = |X|$. Note that $h(X)$, by definition, is a DS function. Assume that $h(X)$ can be expressed as $\frac{f_2(X)}{g_2(X)}$ with $f_2(X)$ and $g_2(X)$ being non-decreasing submodular functions. We then have that $f_2(\emptyset) = f_2(V) = 0$, since $\frac{f_2(\emptyset)}{g_2(\emptyset)} = \frac{f_2(V)}{g_2(V)} = 0$. However, we have that $f_2(\{1\}) = h(\{1\})g_2(\{1\}) > 0$, which contradicts the monotonicity of f_2. □

In Section 3, we give bounded approximation algorithms for RS minimization. This is unlike DS minimization that, in the worst case, is inapproximable (Iyer & Bilmes, 2012b) — hence, we cannot simply optimize $\log f/g$ and expect guarantees. Nevertheless, there are a number of heuristic approaches to DS optimization that work well in practice (Iyer & Bilmes, 2012b; Narasimhan & Bilmes, 2005; Kawahara & Washio, 2011). Moreover, there are several special cases of DS minimization that can be solved exactly (Section 3) and hence where a $(1+\epsilon)$-approximation for RS minimization may be obtained via Algorithm 1.

Relation to SCSC and SCSK: Another related and recently studied class of problems is submodular optimization subject to submodular cover and submodular knapsack constraints (Iyer & Bilmes, 2013), namely:

$$\text{Problem (SCSC): } \min\{f(X) \,|\, g(X) \geq c\}, \qquad (7)$$
$$\text{Problem (SCSK): } \max\{g(X) \,|\, f(X) \leq b\}, \qquad (8)$$

which generalize (Wolsey, 1982; Atamtürk & Narayanan, 2009). These problems are referred to as *Submodular Cost Submodular Cover* (SCSC) and *Submodular Cost Submodular Knapsack* (SCSK), respectively. As shown in (Iyer & Bilmes, 2013), both SCSC and SCSK admit bi-criterion approximation algorithms. Without loss of generality, we concentrate on SCSC. An algorithm is a $[\sigma, \rho]$ bi-criterion algorithm for SCSC if it is guaranteed to obtain a set X such that $f(X) \leq \sigma f(X^*)$ (approximate optimality) and $g(X) \geq \rho c$ (approximate feasibility), where X^* is the optimizer for SCSC. Note that it typically holds that $\sigma \geq 1$ and $\rho \leq 1$. Interestingly, we show in the below that any $[\sigma, \rho]$ bi-criterion algorithm for SCSC can be used as a subroutine to yield a $\frac{\sigma}{\rho}$-approximation algorithm for RS minimization via a simple linear search strategy as described in Algorithm 2.

Algorithm 2 Approx. algorithm for RS minimization using an approximation algorithm for SCSC.

```
1: Input: $f$, $g$, $\epsilon > 0$, and a $[\sigma, \rho]$ bicriterion approximation algorithm for SCSC.
2: Output: An $\frac{\sigma(1+\epsilon)}{\rho}$ approximation for Problem 1.
3: $c \leftarrow g(V)$, $\hat{X}_c \leftarrow V$, and $\hat{X} \leftarrow \hat{X}_c$.
4: while $g(\hat{X}_c) > \min_{j \in V} g(j)$ do
5:    $c \leftarrow (1+\epsilon)^{-1} c$
6:    $\hat{X}_c \leftarrow [\sigma, \rho]$ approx. for Problem 7 with $c$.
7:    if $\frac{f(\hat{X})}{g(\hat{X})} > \frac{f(\hat{X}_c)}{g(\hat{X}_c)}$ then
8:       $\hat{X} \leftarrow \hat{X}_c$
9:    end if
10: end while
11: Return $\hat{X}$.
```

Lemma 2.3. *Given $\epsilon > 0$, Algorithm 2 is guaranteed to find a solution $\hat{X}$ which is a $\frac{\sigma}{\rho}(1+\epsilon)$-approximation for RS minimization in $O(1/\epsilon)$ calls to a $[\sigma, \rho]$ bi-criterion algorithm for SCSC.*

Proof. Let $X^* = \text{argmin}_X \frac{f(X)}{g(X)}$ and $c^* = g(X^*)$. During the linear search procedure, we must have searched a c such that $c \leq c^* \leq (1+\epsilon)c$. For such c, we have that $f(\hat{X}_c) \leq \sigma f(X^*)$ and $g(\hat{X}_c) \geq \rho c$, thanks to the $[\sigma, \rho]$ bi-criterion guarantee. Therefore, we obtain the following: $\frac{f(\hat{X}_c)}{g(\hat{X}_c)} \leq \frac{\sigma}{\rho}\frac{f(X^*)}{c} \leq \frac{\sigma}{\rho}\frac{(1+\epsilon)f(X^*)}{c^*} \leq \frac{\sigma(1+\epsilon)}{\rho}\frac{f(X^*)}{g(X^*)}$. □

Using the same argument, we may connect SCSK to RS maximization via the same linear search strategy. While Lemma 2.3, — showing that a bicriterion approximation algorithm for SCSC (or SCSK) can be utilized to solve RS minimization (maximization) with bounded approximation factors — is theoretically interesting, Algorithm 2 may not be practical for large-scale problems, since it involves solving $O(1/\epsilon)$ instances of SCSC. In Section 3, we provide a number of more efficient approximation algorithms for RS minimization, while still offering similar guarantees.

Algorithm 3 GREEDRATIO for RS minimization

```
1: Input: $f$ and $g$.
2: Output: An approximation solution $\hat{X}$.
3: Initialize: $X_0 \leftarrow \emptyset$, $R \leftarrow V$ and $i \leftarrow 0$.
4: while $R \neq \emptyset$ do
5:     $v \in \text{argmin}_{v \in R} \frac{f(v|X_i)}{g(v|X_i)}$.
6:     $X_{i+1} \leftarrow X_i \cup v$.
7:     $R \leftarrow \{v \in R | g(v|X_{i+1}) > 0\}$
8:     $i \leftarrow i + 1$.
9: end while
10: $i^* \in \text{argmin}_i \frac{f(X_i)}{g(X_i)}$.
11: Return $\hat{X} \leftarrow X_{i^*}$.
```

3. Approximation Algorithms for RS Minimization

In this section, we study four separate cases of RS minimization depending on whether f or g, are modular or submodular.

3.1. Modular f and Modular g

When both f and g are modular, RS minimization becomes very easy. We introduce a simple greedy algorithm– GREEDRATIO (Algorithm 3) to handle this scenario. The idea of GREEDRATIO is to, in each iteration, greedily add an element to the solution set such that the ratio of the marginal gain by this element is the smallest. When the algorithm terminates, a chain of sets $X_1 \subset \ldots, \subset X_\ell$ (ℓ is the total number of iterations) is obtained, and the algorithm simply outputs the set X_{i^*} that achieves the minimum ratio. Though simple, we show below that GREEDRATIO is guaranteed to yield the optimal solution for RS minimization in this case.

Theorem 3.1. *When f and g are modular,* GREEDRATIO *finds the optimal solution for RS minimization with a complexity of $O(n \log n)$.*

Proof. Since both f and g are modular functions, we have that $f(v|X) = f(v)$ and $g(v|X) = g(v)$ for all $X \subseteq V$ and $v \in V \setminus X$. As a simpler implementation of GREEDRATIO, one may first obtain a non-decreasing order of the items in V as $\sigma = \{v_{\sigma_1}, \ldots, v_{\sigma_n}\}$ according to their ratio of singleton scores $\frac{f(v)}{g(v)}$. Namely, $\frac{f(v_{\sigma_1})}{g(v_{\sigma_1})} \leq \cdots \leq \frac{f(v_{\sigma_n})}{g(v_{\sigma_n})}$. It is easy to verify that for any threshold $\tau > 0$, the set $X^\tau = \{v \in V | \frac{f(v)}{g(v)} \leq \tau\}$ is among the chain of solutions $\{X_i\}_{i=1}^n$. Let $X^* \in \text{argmin}_X \frac{f(X)}{g(X)}$ and $r^* = \frac{f(X^*)}{g(X^*)}$. Observe that if any item v satisfies $\frac{f(v)}{g(v)} < r^*$, the item must be contained in X^*, otherwise, adding v to X^* would further decrease the objective. Let $X^{r^*} = \left\{v \in V | \frac{f(v)}{g(v)} < r^*\right\}$. Note that X^{r^*} is contained in the chain of the solutions $\{X_i\}_{i=1}^n$ and that $\frac{f(X^{r^*})}{g(X^{r^*})} = r^*$. Therefore, the output $\hat{X}$ is optimal.

The complexity of algorithm involves computing all n ratio of singleton scores and then takes another $O(n \log n)$ to sort them. □

3.2. Modular f and Submodular g

Next, we study a slightly more general form where f is modular and g is submodular. We show below that this scenario can still be solved up to a constant $1/(1 - 1/e)$ factor by the same greedy algorithm– GREEDRATIO.

Theorem 3.2. *When f is modular and g is submodular,* GREEDRATIO *is guaranteed to obtain a solution $\hat{X}$ such that*

$$\frac{f(\hat{X})}{g(\hat{X})} \leq \frac{e}{e-1} \frac{f(X^*)}{g(X^*)}, \tag{9}$$

where $X^* \in \text{argmin}_{\emptyset \subset X \subseteq V} \frac{f(X)}{g(X)}$.

Proof. This simply follows as a special case of Theorem 3.4 (cf. Section 3.4) when $\kappa_f = 0$. □

We point out that GREEDRATIO may require a time complexity of $O(n^2)$ function evaluations in the worst case. However, thanks to the lazy evaluation trick as described in (Minoux, 1978), Line 5 in GREEDRATIO need not to recompute the marginal gain for every item in each round, allowing GREEDRATIO to scale to large data sets.

3.3. Submodular f and Modular g

In contrast to the case above where f is modular and g is submodular, we show that here, the opposite case, can actually be exactly optimized using the binary search strategy in Algorithm 1. The key observation is that the corresponding DS minimization becomes an instance of submodular minimization, which can be optimally solved in poly-time. Hence, one can achieve a $(1 + \epsilon)$-approximation for this case as a Corollary of Theorem 2.1:

Corollary 3.3. *When f is submodular and g is modular, using an exact submodular minimization algorithm as a subroutine, Algorithm 1 provides a $(1 + \epsilon)$-approximation for RS minimization in $O(\log(1/\epsilon))$ calls to the subroutine.*

3.4. Submodular f and Submodular g

Lastly, we study the most general form of RS minimization with both f and g being submodular. GREEDRATIO can again be shown to yield a curvature dependent bound for this problem. The *curvature* of a submodular function f is defined as follows: (Conforti & Cornuejols, 1984; Vondrák, 2010; Iyer et al., 2013a; Wei et al., 2014):

$$\kappa_f = 1 - \min_{v \in V} \frac{f(v|V \setminus v)}{f(v)} \in [0, 1]. \tag{10}$$

The curvature κ_f measures how close a submodular function f is to a modular function. f is fully curved if $\kappa_f = 1$ and is modular if $\kappa_f = 0$. In the below, we show that GREEDRATIO approximates the RS minimization problem with a factor in terms of the curvature κ_f of the function f.

Theorem 3.4. GREEDRATIO *is guaranteed to obtain a solution $\hat{X}$ such that*

$$\frac{f(\hat{X})}{g(\hat{X})} \leq \frac{1}{1-e^{(\kappa_f-1)}}\frac{f(X^*)}{g(X^*)}, \tag{11}$$

where $X^* \in \text{argmin}_{\emptyset \subset X \subseteq V} \frac{f(X)}{g(X)}$ and κ_f is the curvature of the submodular function f.

Proof. It is equivalent to prove the following:

$$\frac{g(\hat{X})}{f(\hat{X})} \geq (1-e^{(\kappa_f-1)})\frac{g(X^*)}{f(X^*)}. \tag{12}$$

Denote $X_1, X_2, \ldots, X_l$ as the chain of sets obtained by the greedy algorithm and $x_1, \ldots, x_\ell$ as the sequence of items added during the algorithm. Note ℓ is the number of rounds when the algorithm terminates. Denote k as the largest index in $\{1, \ldots, \ell\}$ such that $f(X_k) \leq f(X^*)$. For $i = 1, 2, \ldots, k$, it holds that:

$$\begin{aligned} g(X^*) &\leq g(X_{i-1}) + \sum_{v \in X^* \setminus X_{i-1}} g(v|X_{i-1}) \\ &\leq g(X_{i-1}) + \sum_{v \in X^* \setminus X_{i-1}} \frac{g(x_i|X_{i-1})}{f(x_i|X_{i-1})} f(v|X_{i-1}) \end{aligned}$$

The last inequality follows since $\frac{g(v|X_{i-1})}{f(v|X_{i-1})} \leq \frac{g(x_i|X_{i-1})}{f(x_i|X_{i-1})}$ as required by the greedy algorithm. Given the definition of the curvature κ_f, we have the following

$$\begin{aligned} g(X^*) - g(X_{i-1}) &\leq \frac{g(x_i|X_{i-1})}{f(x_i|X_{i-1})} \sum_{v \in X^* \setminus X_{i-1}} f(v|X_{i-1}) \\ &\leq \frac{g(x_i|X_{i-1})}{f(x_i|X_{i-1})} \sum_{v \in X^*} f(v) \\ &\leq \frac{g(x_i|X_{i-1})}{f(x_i|X_{i-1})} \frac{1}{1-\kappa_f} f(X^*) \end{aligned}$$

Rearranging the inequality, we obtain the following:

$$\frac{g(X^*) - g(X_i)}{g(X^*) - g(X_{i-1})} \leq \left[1 - \frac{(1-\kappa_f)f(x_i|X_{i-1})}{f(X^*)}\right], \tag{13}$$

which implies

$$g(X^*) - g(X_k) \leq \prod_{i=1}^{k}\left[1 - \frac{(1-\kappa_f)f(x_i|X_{i-1})}{f(X^*)}\right] g(X^*) \tag{14}$$

$$\leq \prod_{i=1}^{k} e^{\left[-\frac{(1-\kappa_f)f(x_i|X_{i-1})}{f(X^*)}\right]} g(X^*) \tag{15}$$

$$\leq e^{\left[-\frac{(1-\kappa_f)f(X_k)}{f(X^*)}\right]} g(X^*) \tag{16}$$

Using the fact $1 - x \leq e^{-x}$, we then obtain the following:

$$\frac{g(X_k)}{f(X_k)} \geq \left[1 - e^{\left[-\frac{(1-\kappa_f)f(X_k)}{f(X^*)}\right]}\right] \frac{f(X^*)}{f(X_k)} \frac{g(X^*)}{f(X^*)} \tag{17}$$

$$\geq \left[1 - e^{-(1-\kappa_f)}\right] \frac{g(X^*)}{f(X^*)}. \tag{18}$$

Since $(1 - e^{-(1-\kappa_g)x})x^{-1}$ monotonically decreases in x and $\frac{f(X_k)}{f(X^*)} \leq 1$, we have the following:

$$\frac{f(\hat{X})}{g(\hat{X})} \leq \frac{f(X_k)}{g(X_k)} \leq \frac{1}{1-e^{\kappa_f - 1}} \frac{f(X^*)}{g(X^*)}. \tag{19}$$

□

We point out that Theorem 3.4 generalizes Theorem 3.2 when f is modular, i.e., $\kappa_f = 0$. Note that the approximation guarantee of GREEDRATIO deteriorates as the curvature κ_f of the function f increases and becomes unbounded (and hence vacuous) when the f is fully curved, i.e., $\kappa_f = 1$.

Algorithm 4 ELLIPSOIDAPPROX

```
1: Input: $f$ and $g$
2: Output: An approximation solution $\hat{X}$.
3: $\sqrt{w^f} \leftarrow$ the ellipsoidal approximation for $f$
4: $\hat{X} \in \text{argmin}_X \frac{\sqrt{w^f(X)}}{g(X)}$ \\ $\frac{e(1+\epsilon)}{e-1}$–Approximately solved by Algorithm 2
5: Return $\hat{X}$
```

Ellipsoid Approximation: To yield a bounded approximation algorithm for RS minimization, we provide an algorithmic framework–ELLIPSOIDAPPROX, which involves computing the ellipsoidal approximation (EA) of a submodular function. As shown in Goemans et. al (Goemans et al., 2009), one can approximate any monotone submodular function $f(X)$ in polynomial time by a surrogate function $\hat{f}(X) = \sqrt{w^f(X)}$ for a certain modular weight vector $w^f \in \mathbb{R}^V$, such that

$$\sqrt{w^f(X)} \leq f(X) \leq O(\sqrt{n}\log n)\sqrt{w^f(X)}, \forall X \subseteq V.$$

To apply the idea of EA to RS minimization, we first replace $f(X)$ by its ellipsoidal approximation $\sqrt{w^f(X)}$, and then the problem becomes

$$\min_{\emptyset \subset X \subseteq V} \frac{\sqrt{w^f(X)}}{g(X)}, \tag{20}$$

for which we may use Algorithm 2 (i.e., linear search with SCSC) to solve. At every round of Algorithm 2, the SCSC problem has the following form:

$$\min\{\sqrt{w^f(X)} | g(X) \geq c\}, \tag{21}$$

which is effectively,

$$\min\{w^f(X) | g(X) \geq c\}. \tag{22}$$

Note that Eqn. 22 can be solved by the greedy algorithm with a $[1, (1-1/e)]$ bicriterion approximation guarantee (Iyer & Bilmes, 2013), which then leads to a constant factor approximation for Eqn. 20 thanks to Lemma 2.3. The following result provides a worst-case approximation factor of this approach for RS minimization:

Theorem 3.5. *Let $\sqrt{w^f}$ be the ellipsoidal approximation for f and $\hat{X}$ be the output solution of Algorithm 2 on $\min_{\emptyset \subset X \subseteq V} \frac{\sqrt{w^f(X)}}{g(X)}$, it then holds that*

$$\frac{f(\hat{X})}{g(\hat{X})} \leq O(\sqrt{n}\log n) \frac{f(X^*)}{g(X^*)}. \tag{23}$$

Proof. Let $X^* \in \operatorname{argmin}_X \frac{f(X)}{g(X)}$. Since the output solution $\hat{X}$ of Algorithm 2 has a constant $\frac{e}{e-1}(1+\epsilon)$-approximation for Eqn. 20, it then holds that

$$\frac{f(\hat{X})}{g(\hat{X})} \leq O(\sqrt{n}\log n) \frac{\sqrt{w^f(\hat{X})}}{g(\hat{X})} \tag{24}$$

$$\leq O(\sqrt{n}\log n) \frac{e}{e-1}(1+\epsilon) \frac{\sqrt{w^f(X^*)}}{g(X^*)} \tag{25}$$

$$\leq O(\sqrt{n}\log n) \frac{f(X^*)}{g(X^*)} \tag{26}$$

□

As we will see, the $O(\sqrt{n}\log n)$-approximation factor provided by this approach matches the lower bound (hardness) of the RS minimization up to a log factor.

Majorization-Minimization: While the Ellipsoidal Approximation algorithm provides the tightest approximation factor, it does not scale very well even to medium scale problems (Iyer & Bilmes, 2013). To this end we propose another framework — Majorization-Minimization (MMIN, see Alg. 5) — that achieves a slightly worse approximation factor, but that scales quite well to large scale problems. In the spirit of (Iyer et al., 2013b; Wei et al., 2015b), this framework uses modular lower and modular upper bounds of a submodular function (Iyer et al., 2013b) to transform the originally hard RS minimization problem to a special case with either f or g being modular. For either case, the result admits constant approximation algorithms as shown in Section 3.2 and 3.3. Moreover, the resulting guarantees can be translated to a curvature dependent guarantee for the original RS minimization.

Algorithm 5 MMIN for RS minimization

```
1: Input: $f$ and $g$
2: Output: An approximation solution $\hat{X}$.
3: Initialize: An arbitrary set $X^0$, and $t \leftarrow 0$.
4: repeat
5:     pick a subgradient $h_t$ at $X^t$ of $g$
6:     pick a supergradient $m_t$ at $X^t$ of $f$
7:     $A \in \operatorname{argmin}_X \frac{f(X)}{h_t(X)}$ \\ $(1+\epsilon)-$Approximately solved by Algorithm 1
8:     $B \in \operatorname{argmin}_X \frac{m_t(X)}{g(X)}$ \\ $\frac{e}{e-1}-$Approximately solved by GREEDRATIO
9:     $X^{t+1} \leftarrow \operatorname{argmin}_{X \in \{A,B\}} \frac{f(X)}{g(X)}$
10:    $t \leftarrow t+1$
11: until we have converged ($X^t = X^{t-1}$)
12: Return $\hat{X} \leftarrow X^t$
```

Akin to convex functions, submodular functions have tight modular lower bounds. These bounds are related to the subdifferential $\partial_f(Y)$ of the submodular set function f at a set $Y \subseteq V$, which is defined (Fujishige, 2005) as:

$$\partial_f(Y) = \{y \in \mathbb{R}^n : \tag{27}$$
$$f(X) - y(X) \geq f(Y) - y(Y) \text{ for all } X \subseteq V\}$$

For a vector $x \in \mathbb{R}^V$ and $X \subseteq V$, we write $x(X) = \sum_{j \in X} x(j)$. Denote a subgradient at Y by $h_Y \in \partial_f(Y)$. The extreme points of $\partial_f(Y)$ may be computed via a greedy algorithm: Let π be a permutation of V that assigns the elements in Y to the first $|Y|$ positions ($\pi(i) \in Y$ if and only if $i \leq |Y|$). Each such permutation defines a chain with elements $S_0^\pi = \emptyset$, $S_i^\pi = \{\pi(1), \pi(2), \ldots, \pi(i)\}$ and $S_{|Y|}^\pi = Y$. This chain defines an extreme point h_Y^π of $\partial_f(Y)$ with entries

$$h_Y^\pi(\pi(i)) = f(S_i^\pi) - f(S_{i-1}^\pi). \tag{28}$$

Defined as above, h_Y^π forms a modular lower bound of f, tight at Y — i.e., $h_Y^\pi(X) = \sum_{j \in X} h_Y^\pi(j) \leq f(X), \forall X \subseteq V$ and $h_Y^\pi(Y) = f(Y)$.

We can also define a modular upper bound of a submodular function f via its superdifferentials $\partial^f(Y)$ (Jegelka &

Bilmes, 2011; Iyer & Bilmes, 2012a) at Y:

$$\partial^f(Y) = \{y \in \mathbb{R}^n : \quad f(X) - y(X) \le f(Y) - y(Y); \text{for all } X \subseteq V\} \tag{29}$$

It is possible, moreover, to provide specific supergradients (Iyer & Bilmes, 2012a; Iyer et al., 2013b) that define the following two modular upper bounds (when referring either one, we use m^f_X):

$$m^f_{X,1}(Y) \triangleq f(X) - \sum_{j \in X \setminus Y} f(j|X \setminus j) + \sum_{j \in Y \setminus X} f(j|\emptyset), \tag{30}$$

$$m^f_{X,2}(Y) \triangleq f(X) - \sum_{j \in X \setminus Y} f(j|V \setminus j) + \sum_{j \in Y \setminus X} f(j|X).$$

Then $m^f_{X,1}(Y) \ge f(Y)$ and $m^f_{X,2}(Y) \ge f(Y), \forall Y \subseteq V$ and $m^f_{X,1}(X) = m^f_{X,2}(X) = f(X)$.

Having formally defined the tight modular upper and lower bounds, we are ready to discuss how to apply this machinery to RS minimization. MMIN consists of two stages. In the first stage, it replaces f by its modular upper bound and keep g as it is, and then solves the resulting problem using the algorithms proposed in Section 3.2. In the second stage, MMIN replaces g by its modular lower bound and keep f as it is, and then solves the resulting problem using the algorithms described in Section 3.3. Lastly, MMIN outputs the better among these two solutions. We show below that MMIN yields a bounded approximation for RS minimization in terms of the curvature both of κ_f and κ_g.

Theorem 3.6. MMIN *admits a worst-case approximation factor of* $O(\min\{\frac{n}{1+(n-1)(1-\kappa_f)}, \frac{n}{1+(n-1)(1-\kappa_g)}\})$.

Proof. To prove this theorem, we utilize the Lemma from (Iyer et al., 2013a), that stated the following: Given a monotone submodular function f, it holds that

$$f(X) \le \hat{f}^m(X) \triangleq \sum_{j \in X} f(j) \le \frac{n}{1 + (n-1)(1-\kappa_f)} f(X) \tag{31}$$

where $\hat{f}^m(X)$ is the simple modular upper bound of f. Since, $\hat{f}^m$ approximates f by a factor of $\frac{n}{1+(n-1)(1-\kappa_f)}$, the ratio $\hat{f}^m(X)/g(X)$ also approximates $f(X)/g(X)$ by the same factor. Moreover, the same bound holds for approximating $g(X)$ by its modular lower bound. Hence MMin, produces the two subproblems as discussed in Sections 3.2 and 3.3. Both subproblems admit constant approximation algorithms. □

Observe that Theorem 3.6 provides a worst-case approximation for RS minimization that interpolates between the cases when f and g are modular and submodular. In particular, when either f or g are modular, MMIN provides a constant factor guarantee for this problem.

4. Hardness of RS Optimization

In this section, we provide a worst case hardness result (lower bound) for RS minimization. We show that when f and g are submodular, the problem is NP hard, and one cannot approximate it better than a factor of $O(\sqrt{n})$, which matches the bound of ELLIPSOIDAPPROX up to a log factor.

Theorem 4.1. *There exist an instance of submodular function f and g such that no poly-time algorithm can achieve an approximation factor better than $n^{1/2-\epsilon}$, for any $\epsilon > 0$.*

Proof. We prove this result using the hardness construction from (Goemans et al., 2009; Svitkina & Fleischer, 2008). The main idea of the proof technique is to construct two submodular functions $f(X)$ and $f_R(X)$ that with high probability are indistinguishable. Thus, also with high probability, no algorithm can distinguish between the two functions and the gap in their values provides a lower bound on the approximation.

Define two monotone submodular functions $g(X) = \min\{|X|, \alpha\}$ and $f(X) = \min\{\beta + |X \cap \bar{R}|, |X|, \alpha\}$, where $R \subseteq V$ is a random set of cardinality α. Let α and β be an integer such that $\alpha = x\sqrt{n}/5$ and $\beta = x^2/5$ for an $x^2 = \omega(\log n)$. Given an arbitrary $\epsilon > 0$, set $x^2 = n^{2\epsilon} = \omega(\log n)$. Then the ratio between $g(R)$ and $f(R)$ is $n^{1/2-\epsilon}$. A Chernoff bound analysis very similar to (Svitkina & Fleischer, 2008) reveals that any algorithm that uses a polynomial number of queries can distinguish f and g with probability only $n^{-\omega(1)}$, and therefore cannot reliably distinguish the functions with a polynomial number of queries. Since no algorithm can distinguish between f and g, any algorithm will achieve a minimum value of 1 for the following optimization problem: $\min_X \frac{f(X)}{g(X)}$, whereas the optimal solution has a value $1/n^{1/2-\epsilon}$. □

5. Non-Monotone Submodular and Supermodular f and g

In this section, we investigate extensions of RS minimization to the case when f and g are non-monotone submodular, and even supermodular.

5.1. Non-Monotone Submodular f and g

Given a modular function g and a non-monotone submodular function f, one can use the Binary Search algorithm (Algorithm 1), in which case the corresponding DS subproblem becomes an instance of submodular minimization. Correspondingly, one can easily extend Theorem 2.1 to this case, and achieve a $1 + \epsilon$ approximation factor for this problem. We can also extend our algorithms to the scenario when one of the functions is monotone submodular, while the other one is non-monotone. For example, if f is monotone submodular, while g is non-monotone, one can use Ellipsoidal Approximation on f, and keep g as it is.

The problem then becomes, $\min_X \frac{\sqrt{w_f(X)}}{g(X)}$ which is equivalent to $\max_X \frac{g(X)}{\sqrt{w_f(X)}}$. We can then convert this to SCSK, $\max\{g(X) | \sqrt{w_f(X)} \leq b\}$, which is an instance of non-monotone submodular knapsack, which also has constant factor guarantees (Feige et al., 2011). Furthermore, if f is non-monotone, while g is monotone, one can replace g by its modular upper bound, thereby obtaining an instance of an RS optimization problem, with a non-monotone f and a modular g, which can be solved by using the Binary search algorithm (Algorithm 1) as discussed at the beginning of this section. Finally, in case both f and g are non-monotone submodular (a generalization of (Narasimhan & Bilmes, 2007)), one can use Algorithm 1 and repeatedly solve the resulting DS optimization problem. While this has no guarantees, this is a reasonable heuristic for this problem.

5.2. Extensions to Supermodular f and g

Consider an instance of Problem 1, when f is modular or submodular, but g is supermodular. One can then use Algorithm 1, and the corresponding DS optimization subproblem becomes an instance of submodular minimization, which can exactly solved. Hence in this case Problem 1 is solvable in polynomial time. One can also consider an alternate case, when f is supermodular, and g is either modular or submodular. In this case, the resulting DS optimization problem (using Algorithm 1) becomes an instance of submodular maximization, which can be approximately solved. While this does not directly correspond to an approximation guarantee for the original problem, it does provide a reasonable heuristic for solving the problem. When both f and g are monotone supermodular, it remains an open problem whether it admits any polynomial time algorithm with bounded approximations.

6. Experiments

We empirically evaluate our proposed algorithmic frameworks for RS minimization, including in particular MMIN, GREEDRATIO, and ELLIPSOIDAPPROX, on a synthetic data set. In particular, we evaluate on a generalized form of the F-measure function:

$$F_\lambda(X) = \frac{|\Gamma(X) \cap T|}{\lambda|T| + (1-\lambda)|\Gamma(X)|}, \tag{32}$$

where $0 \leq \lambda \leq 1$ is a parameter that determines a trade-off weight between precision and recall. Note $F_{\lambda=0.5}$ is the same as the F-measure function defined in Eqn. 3. We instantiate the F-measure function on a randomly generated bipartite graph $G(U, W, E)$. The bipartite graph is defined with $|U| = 100$ and $|W| = 100$. We define an edge between $u \in U$ and $w \in W$ independently with probability $p = 0.05$. The set of targets $T \subseteq W$ is also randomly chosen with a fixed size 20, i.e., $|T| = 20$. We run the experiments on 10 instances of the randomly and independently generated data, and we report the averaged results.

As a baseline, we also implement a random sampling method, where we randomly choose 100 subsets $X \subseteq U$ with size $|X| = 50$ and report their averaged function valuation in terms of F_λ and their standard deviation. In Figure 1, we compare the performance of all methods on with the varying λ. We observe that GREEDRATIO, though very efficient, achieves consistently the best performance for all cases of λ among all optimization algorithms. Comparable performance is achieved by MMIN and ELLIPSOIDAPPROX, although ELLIPSOIDAPPROX is much more computationally intensive.

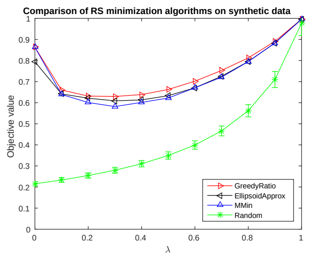

Figure 1. synthetic data experiment

Acknowledgments

We wish acknowledge support by the National Science Foundation under Grant No. IIS-1162606, the National Institutes of Health under award R01GM103544, by a Google, a Microsoft, and an Intel research award, and also support by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

Atamtürk, Alper and Narayanan, Vishnu. The submodular knapsack polytope. *Discrete Optimization*, 2009.

Conforti, M. and Cornuejols, G. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Applied Mathematics*, 7(3):251–274, 1984.

Feige, Uriel, Mirrokni, Vahab, and Vondrák, Jan. Maximizing non-monotone submodular functions. *SIAM J. COMPUT.*, 40(4):1133–1155, 2011.

Fujishige, S. *Submodular functions and optimization*, volume 58. Elsevier Science, 2005.

Goemans, M.X., Harvey, N.J.A., Iwata, S., and Mirrokni, V. Approximating submodular functions everywhere. In *SODA*, pp. 535–544, 2009.

Iyer, R. and Bilmes, J. The submodular Bregman and Lovász-Bregman divergences with applications. In *NIPS*, 2012a.

Iyer, R. and Bilmes, J. Algorithms for approximate minimization of the difference between submodular functions, with applications. *In UAI*, 2012b.

Iyer, R. and Bilmes, J. Submodular Optimization with Submodular Cover and Submodular Knapsack Constraints. In *NIPS*, 2013.

Iyer, R., Jegelka, S., and Bilmes, J. Curvature and Optimal Algorithms for Learning and Minimizing Submodular Functions . In *Neural Information Processing Society (NIPS)*, 2013a.

Iyer, R., Jegelka, S., and Bilmes, J. Fast Semidifferential based Submodular function optimization. In *ICML*, 2013b.

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

Kawahara, Yoshinobu and Washio, Takashi. Prismatic algorithm for discrete dc programming problems. In *NIPS*, 2011.

Kawahara, Yoshinobu, Nagano, Kiyohito, and Okamoto, Yoshio. Submodular fractional programming for balanced clustering. *Pattern Recognition Letters*, 32(2):235–243, 2011.

Krause, A., Singh, A., and Guestrin, C. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, 9:235–284, 2008.

Lin, H. and Bilmes, J. Optimal selection of limited vocabulary speech corpora. In *Interspeech*, 2011.

Liu, Yuzong, Wei, Kai, Kirchhoff, Katrin, Song, Yisong, and Bilmes, Jeff. Submodular feature selection for high-dimensional acoustic score spaces. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 7184–7188. IEEE, 2013.

Lovász, L. Submodular functions and convexity. *Mathematical Programming*, 1983.

Minoux, M. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, pp. 234–243, 1978.

Narasimhan, M. and Bilmes, J. A submodular-supermodular procedure with applications to discriminative structure learning. In *UAI*, 2005.

Narasimhan, Mukund and Bilmes, Jeff. Local search for balanced submodular clusterings. In *IJCAI*, pp. 981–986, 2007.

Shi, Jianbo and Malik, Jitendra. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

Svitkina, Z. and Fleischer, L. Submodular approximation: Sampling-based algorithms and lower bounds. In *FOCS*, pp. 697–706, 2008.

Vondrák, J. Submodularity and curvature: the optimal algorithm. *RIMS Kokyuroku Bessatsu*, 23, 2010.

Wei, Kai, Liu, Yuzong, Kirchhoff, Katrin, and Bilmes, Jeff. Using document summarization techniques for speech data subset selection. In *NAACL-HLT*, 2013.

Wei, Kai, Iyer, Rishabh, and Bilmes, Jeff. Fast multi-stage submodular maximization. Beijing, China, 2014.

Wei, Kai, Iyer, Rishabh, and Bilmes, Jeff. Submodularity in data subset selection and active learning. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1954–1963, 2015a.

Wei, Kai, Iyer, Rishabh K, Wang, Shengjie, Bai, Wenruo, and Bilmes, Jeff A. Mixed robust/average submodular partitioning: Fast algorithms, guarantees, and applications. In *Advances in Neural Information Processing Systems*, pp. 2224–2232, 2015b.

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