Curvature and Optimal Algorithms for Learning and Minimizing Submodular Functions

Authored by:

Jeff A. Bilmes
Stefanie Jegelka
Rishabh K. Iyer

Abstract

We investigate three related and important problems connected to machine learning, namely approximating a submodular function everywhere, learning a submodular function (in a PAC like setting [26]), and constrained minimization of submodular functions. In all three problems, we provide improved bounds which depend on the ?curvature? of a submodular function and improve on the previously known best results for these problems [9, 3, 7, 25] when the function is not too curved ? a property which is true of many real-world submodular functions. In the former two problems, we obtain these bounds through a generic black-box transformation (which can potentially work for any algorithm), while in the case of submodular minimization, we propose a framework of algorithms which depend on choosing an appropriate surrogate for the submodular function. In all these cases, we provide almost matching lower bounds. While improved curvature-dependent bounds were shown for monotone submodular maximization [4, 27], the existence of similar improved bounds for the aforementioned problems has been open. We resolve this question in this paper by showing that the same notion of curvature provides these improved results. Empirical experiments add further support to our claims.

1 Paper Body

We investigate three related and important problems connected to machine learning: approximating a submodular function everywhere, learning a submodular function (in a PAC-like setting [28]), and constrained minimization of submodular functions. We show that the complexity of all three problems depends on the ?curvature? of the submodular function, and provide lower and upper bounds that refine and improve previous results [2, 6, 8, 27]. Our proof techniques are fairly generic. We either use a black-box transformation of the function (for approximation and learning), or a transformation of algorithms to use an appropriate surrogate function (for minimization). Curiously, curvature

has been known to influence approximations for submodular maximization [3, 29], but its effect on minimization, approximation and learning has hitherto been open. We complete this picture, and also support our theoretical claims by empirical results.

1

Introduction

Submodularity is a pervasive and important property in the areas of combinatorial optimization, economics, operations research, and game theory. In recent years, submodularity?s use in machine learning has begun to proliferate as well. A set function f : 2V ? R over a finite set V = {1, 2, . . . , n} is submodular if for all subsets S, T ? V , it holds that f (S) + f (T) ? f (S ? T) + / S in the context S as f (S ? T). Given a set S ? V , we define the gain of an element j ? f (j—S) , f (S ? j) ? f (S). A function f is submodular if it satisfies diminishing marginal returns, namely f (j—S) ? f (j—T) for all S ? T, j ? / T , and is monotone if f (j—S) ? 0 for all j ? / S, S ? V . While submodularity, like convexity, occurs naturally in a wide variety of problems, recent studies have shown that in the general case, many submodular problems of interest are very hard: the problems of learning a submodular function or of submodular minimization under constraints do not even admit constant or logarithmic approximation factors in polynomial time [2, 7, 8, 10, 27]. These rather pessimistic results however stand in sharp contrast to empirical observations, which suggest that these lower bounds are specific to rather contrived classes of functions, whereas much better results can be achieved in many practically relevant cases. Given the increasing importance of submodular functions in machine learning, these observations beg the question of qualifying and quantifying properties that make sub-classes of submodular functions more amenable to learning and optimization. Indeed, limited prior work has shown improved results for constrained minimization and learning of sub-classes of submodular functions, including symmetric functions [2, 25], concave functions [7, 18, 24], label cost or covering functions [9, 31]. In this paper, we take additional steps towards addressing the above problems and show how the generic notion of the curvature ? the deviation from modularity? of a submodular function determines both upper and lower bounds on approximation factors for many learning and constrained optimization problems. In particular, our quantification tightens the generic, function-independent bounds in [8, 2, 27, 7, 10] for many practically relevant functions. Previously, the concept of curvature has been used to 1

tighten bounds for submodular maximization problems [3, 29]. Hence, our results complete a unifying picture of the effect of curvature on submodular problems. Moreover, curvature is still a fairly generic concept, as it only depends on the marginal gains of the submodular function. It allows a smooth transition between the ?easy? functions and the ?really hard? subclasses of submodular functions.

2

Problem statements, definitions and background

Before stating our main results, we provide some necessary definitions and

introduce a new concept, the curve normalized version of a submodular function. Throughout this paper, we assume that the submodular function f is defined on a ground set V of n elements, that it is nonnegative and V f (?) = 0. We also use normalized modular (or P additive) functions w : 2 ? R which are those that can be written as a sum of weights, w(S) = i?S w(i). We are concerned with the following three problems: Problem 1. (Approximation [8]) Given a submodular function f in form of a value oracle, find an approximation f? (within polynomial time and representable within polynomial space), such that for all X ? V , it holds that f?(X) ? f (X) ? ?1 (n)f?(X) for a polynomial ?1 (n). Problem 2. (PMAC-Learning [2]) Given i.i.d training samples {(Xi , f (Xi)}m i=1 from a distribution D, learn an approximation f?(X) that is, with probability 1 ? ?, within a multiplicative factor of ?2 (n) from f . Problem 3. (Constrained optimization [27, 7, 10, 16]) Minimize a submodular function f over a family C of feasible sets, i.e., minX?C f (X). In its general form, the approximation problem was first studied by Goemans et al. [8], who approx? imate any monotone submodular function to within a factor of O(n log n), with a lower bound ? of ?1 (n) = ?(n/ log n). Building on this result, Balcan and Harvey ? [2] show how to PMAC-learn a monotone submodular function within a factor of ?2 (n) = O(n), and prove a lower bound of ?(n1/3) for the learning problem. Subsequent work extends these results to sub-additive and fractionally sub-additive functions [1]. Better learning results are possible for the subclass of submodular shells [23] and Fourier sparse set functions [26]. Both Problems 1 and 2 have numerous applications in algorithmic game theory and economics [2, 8] as well as machine learning [2, 22, 23, 26, 15]. Constrained submodular minimization arises in applications such as power assignment or transportation problems [19, 30, 13]. In machine learning, it occurs, for instance, in the form of MAP inference in high-order graphical models [17] or in size-constrained corpus extraction [21]. Recent results show that almost all constraints make it hard to solve the minimization even within a constant factor [27, 6, 16]. Here, we will focus on the constraint of imposing a lower bound on the cardinality, and on combinatorial constraints where C is the set of all s-t paths, s-t cuts, spanning trees, or perfect matchings in a graph. A central concept in this work is the total curvature ?f of a submodular function f and the curvature ?f (S) with respect to a set S ? V , defined as [3, 29] ?f = 1 ? min j?V

f (j — V j) , f (j)

?f (S) = 1 ? min j?S

f (j—Sj) . f (j)

(1)

Without loss of generality, assume that f (j) ¿ 0 for all j ? V . It is easy to see that ?f (S) ? ?f (V) = ?f , and hence ?f (S) is a tighter notion of curvature. A modular function has curvature ?f = 0, and a matroid rank function has maximal curvature ?f = 1. Intuitively, ?f measures how far away f is from being modular. Conceptually, curvature is distinct from the recently proposed submodularity ratio [5] that measures how far a function is from being submodular. Curvature has served to tighten bounds for submodular maximization problems, e.g., from (1?1/e) to ?1f (1?e??f) for monotone submodular maximization subject to a

cardinality constraint [3] or matroid constraints [29], and these results are tight. For submodular minimization, learning, and approximation, however, the role of curvature has not yet been addressed (an exception are the upper bounds in [13] for minimization). In the following sections, we complete the picture of how curvature affects the complexity of submodular maximization and minimization, approximation, and learning. The above-cited lower bounds for Problems 1?3 were established with functions of maximal curvature (?f = 1) which, as we will see, is the worst case. By contrast, many practically interesting functions have smaller curvature, and our analysis will provide an explanation for the good empirical results 2

observed with such functions [13, 22, 14]. An example for functions with ?f ¡ 1 is the class of concave over modular functions that have been used in speech processing [22] and computer Pk vision [17]. This class comprises, for instance, functions of the form f (X) = i=1 (wi (X))a , for some a ? [0, 1] and a nonnegative weight vectors wi . Such functions may be defined over clusters Ci ? V , in which case the weights wi (j) are nonzero only if j ? Ci [22, 17, 11]. Curvature-dependent analysis. To analyze Problems 1 ? 3, we introduce the concept of a curvenormalized polymatroid1 . Specifically, we define the ?f -curve-normalized version of f as P f (X) ? (1 ? ?f) j?X f (j) ? f (X) = (2) ?f If ?f = 0, then we set f ? ? 0. We call f ? the curve-normalized version of f because its curvature is ?f ? = 1. The function f ? allows us to decompose a submodular function f into a ?difficult? polymatroid function and an ?easy? modularP part as f (X) = fdifficult (X) + measy (X) where fdifficult (X) = ?f f ? (X) and measy (X) = (1 ? ?f) j?X f (j). Moreover, we may modulate the curvature of given any function g with ?g = 1, by constructing a function f (X) , cg(X) + (1 ? c)—X— with curvature ?f = c but otherwise the same polymatroidal structure as g. Our curvature-based decomposition is different from decompositions such as that into a totally normalized function and a modular function [4]. Indeed, the curve-normalized function has some specific properties that will be useful later on (proved in [12]): P Lemma 2.1. If f is monotone submodular with ?f ¿ 0, then f (X) ? j?X f (j) and f (X) ? P (1 ? ?f) j?X f (j). ? Lemma 2.2. If f is monotone submodular, then P f (X) in Eqn. (2) is a monotone non-negative ? submodular function. Furthermore, f (X) ? j?X f (j).

The function f ? will be our tool for analyzing the hardness of submodular problems. Previous information-theoretic lower bounds for Problems 1?3 [6, 8, 10, 27] are independent of curvature and use functions with ?f = 1. These curvature-independent bounds are proven by constructing two essentially indistinguishable matroid rank functions h and f R , one of which depends on a random set R ? V . One then argues that any algorithm would need to make a super-polynomial number of queries to the functions for being able to distinguish h and f R with high enough probability. The lower bound will be the ratio maxX?C h(X)/f R (X). We extend this proof technique to functions with a fixed given curvature. To this end, we define the functions f?R (X) = ?f f R (X) + (1 ? ?f)—X—

and

h? (X) = ?f h(X) + (1 ? ?f)—X—.

(3)

Both of these functions have curvature ?f . This construction enables us to explicitly introduce the effect of curvature into information-theoretic bounds for all monotone submodular functions. Main results. The curve normalization (2) leads to refined upper bounds for Problems 1?3, while the curvature modulation (3) provides matching lower bounds. The following are some of our main results: for approximating submodular functions (Problem 1), we replace the known bound ? ? n log n). We of ?1 (n) = O(n log n) [8] by an improved curvature-dependent O(1+(?n log n?1)(1?? f) ? n ? complement this with a lower bound of ?(1+(?n?1)(1??f)). For learning submodular functions ? (Problem 2), we refine the known bound of ?2 (n) = O(n) [2] in the PMAC setting to a curvature ? n n1/3 ? dependent bound of O(1+(?n?1)(1??), with a lower bound of ?( ). Finally, 1+(n1/3 ?1)(1??f) f) Table 1 summarizes our curvature-dependent approximation bounds for constrained minimization (Problem 3). These bounds refine many of the results in [6, 27, 10, 16]. In general, our new curvaturedependent upper and lower bounds refine known theoretical results whenever ?f ¡ 1, in many cases replacing known polynomial bounds by a curvature-dependent constant factor 1/(1 ? ?f). Besides making these bounds precise, the decomposition and the curve-normalized version (2) are the basis for constructing tight algorithms that (up to logarithmic factors) achieve the lower bounds. 1

A polymatroid function is a monotone increasing, nonnegative, submodular function satisfying f (?) = 0.

3

Constraint

Modular approx. (MUB)

Card. LB

k 1+(k?1)(1??f) n 1+(n?1)(1??f) n 2+(n?2)(1??f) n 1+(n?1)(1??f) m 1+(m?1)(1??f)

Spanning Tree Matchings s-t path s-t cut

Ellipsoid approx. (EA)

? log n O(1+(?n lognn?1)(1??) f) ? m log m ? O(1+(m log m?1)(1??f)) ? m log m O(1+(?m log) ? m?1)(1??f) m log m O(1+(?m log m?1)(1??f)) ? ? log m) O(1+(log mm m?1)(1??f)

Lower bound 1/2) ?) ?(1+(n1/2n?1)(1?? f) n ? ?( ) 1+(n?1)(1??f)

n ? ?(1+(n?1)(1??f)) 2/3 n ?) ?(2/3 1+(n

?1)(1??f) ?

n ? ?) ?(1+(n?1)(1??f)

Table 1: Summary of our results for constrained minimization (Problem 3).

3

Approximating submodular functions everywhere

We first address improved bounds for the problem of approximating a monotone submodular function everywhere. Previous work established ?-approximations g to a submodular function f satisfying g(S) ? f (S) ? ?g(S) for all S ? V [8]. We begin with a theorem showing how any algorithm computing such an approximation may be used to obtain a curvature-specific, improved approximation.

Note that the curvature of a monotone submodular function can be obtained within 2n + 1 queries to f . The key idea of Theorem 3.1 is to only approximate the curved part of f , and to retain the modular part exactly. The full proof is in [12]. Theorem 3.1. Given a polymatroid function f with ?f ¡ 1, let f ? be its curve-normalized version defined in Equation (2), and let f?? be a submodular function satisfying f?? (X) ? f ? (X) ? P ?(n)f?? (X), for some X ? V . Then the function f?(X) , ?f f?? (X) + (1 ? ?f) j?X f (j) satisfies f?(X) ? f (X) ?

?(n) f?(X) f?(X) ? . 1 + (?(n) ? 1)(1 ? ?f) 1 ? ?f

(4)

Theorem 3.1 may be directly applied to tighten recent results on approximating submodular functions everywhere. An algorithm by Goemans et al. [8] computes an approximation to a polymatroid function f in polynomial time by approximating the submodular polyhedron via?an ellipsoid. This approximation p (which we call the ellipsoidal approximation) satisfies ?(n) = O(n log n), and has the form wf (X) for a certain weight vector wf . Corollary 3.2 states that a tighter approximation is possible for functions with ?f ¡ 1. p Corollary 3.2. Let f be a polymatroid function with ?f ¡ 1, and let wf ? (X) be the ellipsoidal to the ?-curve-normalized version f ? (X) of f . Then the function f ea (X) = p approximation P ? f ?f w (X) + (1 ? ?f) j?X f (j) satisfies

? n log n ? f ea (X) ? f (X) ? O f ea (X). (5) 1 + (n log n ? 1)(1 ? ?f) If ?f = 0, then the approximation is exact. This is not surprising since a modular function can be inferred exactly within O(n) oracle calls. The following lower bound (proved in [12]) shows that Corollary 3.2 is tight up to logarithmic factors. It refines the lower bound in [8] to include ?f . Theorem 3.3. Given a submodular function f with curvature ?f , there does not exist a (possibly randomized) polynomial-time algorithm that computes an approximation to f within a factor of n1/2? , for any ¿ 0. 1+(n1/2? ?1)(1??f) ?m The P simplest alternative approximation to f one might conceive is the modular function f (X) , f (j) which can easily be computed by querying the n values f (j). j?X Lemma 3.1. Given a monotone submodular function f , it holds that2 X —X— f (X) ? f?m (X) = f (j) ? f (X) 1 + (—X— ? 1)(1 ? ?f (X)) j?X

2

In [12], we show this result with a stronger notion of curvature: ??f (X) = 1 ?

4

P

j?X P

f (j—Xj) . f (j)

j?X

(6)

The form of Lemma 3.1 is slightly different from Corollary 3.2. However, there is a straightforward correspondence: given f? such that f?(X) ? f (X) ? ?0 (n)f?(X), by defining f?0 (X) = ?0 (n)f?(X), we get that f (X) ? f?0 (X) ? ?0 (n)f (X). Lemma 3.1 for the modular approximation ? is complementary to Corollary 3.2: First, the modular approximation is better whenever —X— ? n. Second, the bound in Lemma 3.1 depends on the curvature ?f (X) with respect to the set

X, which is stronger than ?f . Third, f?m is extremely simple to compute. For sets of larger cardinality, however, the ellipsoidal approximation of Corollary 3.2 provides a better approximation, in fact, the best possible one (Theorem 3.3). In a similar manner, Lemma 3.1 is tight for any modular approximation to a submodular function: Lemma 3.2. For any ? ¿ 0, there exists a monotone submodular function f with curvature ? such —X— that no modular upper bound on f can approximate f (X) to a factor better than 1+(—X—?1)(1?? . f) The improved curvature dependent bounds immediately imply better bounds for the class of concave over modular functions used in [22, 17, 11]. Corollary 3.4. Given weight vectors w1 , ? ? ? , wk ? 0, and a submodular function f (X) = Pk P a 1?a f (X) i=1 ?i [wi (X)] , ?i ? 0, for a ? (0, 1), it holds that f (X) ? j?X f (j) ? —X— In particular, when a = 1/2, thep modular upper bound approximates the sum of square-root over modular functions by a factor of —X—.

4

Learning Submodular functions

We next address the problem of learning submodular functions in a PMAC setting [2]. The PMAC (Probably Mostly Approximately Correct) framework is an extension of the PAC framework [28] to allow multiplicative errors in the function values from a fixed but unknown distribution D over 2V . We are given training samples {(Xi , f (Xi)}m i=1 drawn i.i.d. from D. The algorithm may take time polynomial in n, 1/, 1/? to compute a (polynomially-representable) function f? that is a good approximation to f with respect to D. Formally, f? must satisfy that h i PrX1 ,X2 ,??? ,Xm ?D PrX?D [f?(X) ? f (X) ? ?(n)f?(X)] ? 1 ? ? 1 ? ? (7) for some approximation factor ?(n). Balcan and Harvey [2] propose an algorithm that PMAC-learns ? any monotone, nonnegative submodular function within a factor ?(n) = n + 1 by reducing the problem to that of learning a binary classifier. If we assume that we have an upper bound on the curvature ?f , or that we can estimate it 3 , and have access to the value of the singletons f (j), j ? V , then we can obtain better learning results with non-maximal curvature: Lemma 4.1. Let f be a monotone submodular function for which we know an upper bound on its curvature and the singleton weights f (j) for all j ? V . For every , ? ¿ 0 there is an algorithm that uses a polynomial number of training examples, runs in time polynomial in (n, 1/, 1/?) and ? n+1 PMAC-learns f within a factor of 1+(?n+1?1)(1?? . If D is a product distribution, then there exists) f

log

1

an algorithm that PMAC-learns f within a factor of O(1+(log 1 ?1)(1??).)

f

The algorithm of Lemma 4.1 uses the reduction of Balcan and Harvey [2] to learn the ?f -curvenormalized version f ? of f . From the learned function f?? (X), we construct the final estimate P f?(X) , ?f f?? (X) + (1 ? ?f) j?X f (j). Theorem 3.1 implies Lemma 4.1 for this f?(X). Moreover, no polynomial-time algorithm can be guaranteed to PMAC-learn f within a factor of 0 n1/3? , for

any 0 ¿ 0 [12]. We end this section by showing how we can learn with a 0 1/3? 1+(n ?1)(1??f) construction analogous to that in Lemma 3.1. Lemma 4.2. If f is a monotone submodular function with known curvature (or a known upper bound) ??f (X), ?X ? V , then for every , ? ¿ 0 there is an algorithm that uses a polynomial number of training examples, runs in time polynomial in (n, 1/, 1/?) and PMAC learns f (X) within a factor —X— of 1 + 1+(—X—?1)(1? ??f (X)) . 3 note that ?f can be estimated from a set of 2n + 1 samples {(j, f (j))}j?V , {(V, f (V))}, and {(V j, f (V j)}j?V included in the training samples

5

Compare this result to Lemma 4.1. Lemma 4.2 leads to better bounds for small sets, whereas Lemma 4.1 provides a better general bound. Moreover, in contrast to Lemma 4.1, here we only need an upper bound on the curvature and do not need to know the singleton weights {f (j), j ? V }. Note also that, while ?f itself is an upper bound of ??f (X), often one does have an upper bound on ??f (X) if one knows the function class of f (for example, say concave over modular). In particular, an immediate Pk corollary is that the class of concave over modular = i=1 ?i [wi (X)]a , ?i ? 0, for ? functions f (X) a ? (0, 1) can be learnt within a factor of min{ n + 1, 1 + —X—1?a }.

5

Constrained submodular minimization

Next, we apply our results to the minimization of submodular functions under constraints. Most algorithms for constrained minimization use one of two strategies: they apply a convex relaxation [10, 16], or they optimize a surrogate function f? that should approximate f well [6, 8, 16]. We follow the second strategy and propose a new, widely applicable curvature-dependent choice for surrogate functions. A suitable selection of f? will ensure theoretically optimal results. Throughout this section, we refer to the optimal solution as X ? ? argminX?C f (X). Lemma 5.1. Given a submodular function f , let f?1 be an approximation of f such that f?1 (X) ? b1 ? argminX?C f?(X) of f? satisfies f (X) ? ?(n)f?1 (X), for all X ? V . Then any minimizer X b ? ?(n)f (X ?). Likewise, if an approximation of f is such that f (X) ? f?2 (X) ? ?(X)f (X) f (X) ? 2 ? argminX?C f?2 (X) satisfies f (X b2) ? for a set-specific factor ?(X), then its minimizer X ? ? 4 ? ? ?(X)f (X). If only ?-approximations are possible for minimizing f1 or f2 over C, then the final bounds are ??(n) and ??(X ?) respectively. For Lemma 5.1 to be practically useful, it is essential that f?1 and f?2 be efficiently optimizable over C. We discuss two general curvature-dependent approximations that work for a large class of combinatorial constraints. In particular, we use Theorem 3.1: we decompose f into f ? and a modular part f m , and then approximate f ? while retaining f m , i.e., f? = f?? + f m . The first approach uses a simple modular upper bound (MUB) and the second relies on the Ellipsoidal approximation (EA) we used in Section 3. MUB: The simplest approximation to a submodular function is the modular approximation P f?m (X) , j?X f (j) ? f (X). Since here, f?? happens to be equivalent to f m , we obtain the overall approximation f? = f?m . Lemmas 5.1 and 3.1 directly imply a set-dependent approximation factor for f?m : b ? C be a ?-approximate solution for minimizing P Corollary 5.1. Let X j?X f (j)

over C, i.e. P P f (j) ? ? min f (j). Then X?C b j?X j?X ? ? f (X)

1+

(—X ? —

?—X ? — f (X ?). ? 1)(1 ? ?f (X ?))

(8)

Corollary 5.1 has also been shown in [13]. Similar to the algorithms in [13], MUB can be extended to an iterative algorithm yielding performance gains in practice. In particular, Corollary 5.1 implies improved approximation bounds for practically relevantq concave over modular functions, such Pk P as those used in [17]. For instance, for f (X) = i=1 j?X wi (j), we obtain a worst-case p ? approximation bound of —X ? — ? n. This is significantly better than the worst case factor of —X ? — for general submodular functions. ? EA: Instead of employing a modular upper bound, we can approximate p f? using the construction f ? by Goemans et al. [8], as in Corollary 3.2. In that case, f (X) = ?f w (X) + (1 ? ?f)f m (X) has a special form: a weighted sum of a concave function and a modular function. Minimizing such a function over constraints C is harder than minimizing a merely modular function, but with the algorithm in [24] we obtain an FPTAS5 for minimizing f? over C whenever we can minimize a nonnegative linear function over C. 4 5

A ?-approximation algorithm for minimizing a function g finds set X : g(X) ? ? minX?C g(X) The FPTAS will yield a ? = (1 +)-approximation through an algorithm polynomial in 1 .

6

Corollary 5.2. For a submodular function with curvature ?f ¡ 1, algorithm EA will return a b that satisfies solution X

? n log n b f (X) ? O ? f (X ?). (9) (n log n ? 1)(1 ? ?f) + 1) Next, we apply the results of this section to specific optimization problems, for which we show (mostly tight) curvature-dependent upper and lower bounds. We just state our main results; a more extensive discussion along with the proofs can be found in [12]. Cardinality lower bounds (SLB). A simple constraint is a lower bound on the cardinality of the solution, i.e., C = {X ? V : —X— ? k}. Svitkina and Fleischer [27] prove that for monotone submodular functions of arbitrary curvature, p it is impossible to find a polynomial-time algorithm with an approximation factor better than n/ log n. They show an algorithm which matches this approximation factor. Corollaries 5.1 and?5.2 immediately imply curvature-dependent approximation log n k bounds of 1+(k?1)(1?? and O(1+(?n lognn?1)(1??). These bounds are improvements over the f) f) results of [27] whenever ?f ¡ 1. Here, MUB is preferable to EA whenever k is small. Moreover, the bound of EA is tight up to poly-log factors, in that no polynomial time algorithm can achieve a n1/2? general approximation factor better than 1+(n1/2? for any ¿ 0. ?1)(1??f) In the following problems, our ground set V consists of the set of edges in a graph G = (V, E) with two distinct nodes s, t ? V and n = —V—, m = —E—. The submodular function is f : 2E ? R. Shortest submodular s-t path (SSP). Here, we aim to find an s-t path X of minimum (submodular) length f (X). Goel et al. [6] show a O(n2/3)-approximation with matching curvature-independent lower bound ?(n2/3). By Corollary 5.1, the

curvature-dependent worst-case bound for MUB is n 1+(n?1)(1??f) since any minimal s-t path has at most n edges. Similarly, the factor for EA is ?

m log m O(1+(?m log). The bound of EA will be tighter for sparse graphs while MUB provides m?1)(1??f)

better results for dense ones. Our curvature-dependent lower bound for SSP is for any ¿ 0, which reduces to the result in [6] for ?f = 1.

n2/3? , 1+(n2/3? ?1)(1??f)

Minimum submodular s-t cut (SSC): This problem, also known as the cooperative cut problem [16, 17], asks to minimize a monotone submodular function f such that the solution X ? E is a set of edges whose removal disconnects s from t in G. Using curvature refines the We can also show a n1/2? lower bound of [16] to 1+(n1/2? , for any ¿ 0. Corollary 5.1 implies an approximation ?1)(1??f) ?

m log m m factor of O((?m log m?1)(1??) for EA and a factor of 1+(m?1)(1?? for MUB, where m = —E— f) f)+1 is the number of edges in the graph. Hence the factor for EA is tight for sparse graphs. Specifically for cut problems, there is yet another useful surrogate function that is exact on local neighborhoods. Jegelka and Bilmes [16] demonstrate how this approximation may be optimized via a generalized maximum flow algorithm that maximizes a polymatroidal network flow [20]. This algorithm still applies to the combination f? = ?f f?? + (1 ? ?f)f m , where we only approximate f ? . We refer to this approximation as Polymatroidal Network Approximation (PNA). n Corollary 5.3. Algorithm PNA achieves a worst-case approximation factor of 2+(n?2)(1?? for the f) cooperative cut problem.

For dense graphs, this factor is theoretically tighter than that of the EA approximation. Minimum submodular spanning tree (SST). Here, C is the family of all spanning trees in a given graph G. Such constraints occur for example in power assignment problems [30]. Goel et al. [6] show a curvature-independent optimal approximation factor of O(n) for this problem. Corollary 5.1 n refines this bound to 1+(n?1)(1?? when using MUB; Corollary 5.2 implies a slightly worse bound f) for EA. We also show that the bound of MUB is tight: no polynomial-time algorithm can guarantee a n1? , for any , ? ¿ 0. factor better than 1+(n1? ?1)(1?? f)+??f Minimum submodular perfect matching (SPM): Here, we aim to find a perfect matching in a graph that minimizes a monotone submodular function. Corollary 5.1 implies that an MUB n approximation will achieve an approximation factor of at most 2+(n?2)(1?? . Similar to the f) spanning tree case, the bound of MUB is also tight [12]. 7

4 2 50

100 150 200 250 n

?= 0.7 ?= 0.5 ?= 0.3 ?= 0.1

4 3 2 50

100 150 200 250 n

(c) with varying ? and ? = 1 100 80 60

?= n/2 ?= n3/4 1/2

?= n

40 20 50

100 n
150
200
(d) varying ? with ? = n/2 and ? = 1 emp. approx. factor
6
(b) varying ?, ?= 0.1 5
emp. approx. factor
?= 0.1 ?= 0.2 ?= 0.3 ?= 0.4
emp. approx. factor
emp. approx. factor
(a) varying ?, ?= 0 8
?= 0.9 ?= 0.6 ?= 0.3 ?= 0.1
8 6 4 2 50
100 n
150
200

Figure 1: Minimization of g? for cardinality lower bound constraints. (a) fixed ? = 0, ? = n1/2+ , ? = n2 for varying ; (b) fixed = 0.1, but varying ?; (c) different choices of ? for ? = 1; (d) varying ? with ? = n/2, ? = 1. Dashed lines: MUB, dotted lines: EA, solid lines: theoretical bound. The results of EA are not visible in some instances since it obtains a factor of 1. 5.1

Experiments

We end this section by empirically demonstrating the performance of MUB and EA and their precise dependence on curvature. We focus on cardinality lower bound constraints, C = {X ? V : —X— ? ?} and the ?worst-case? class of functions that has been used throughout this paper to prove lower ? = V R and R ? V is random set such that ? + ?, —X—, ?} where R bounds, f R (X) = min{—X ? R— 1/2+ 2 —R— = ?. We adjust ? = n and ? = n by a parameter . The smaller is, the harder the problem. This function has curvature ?f = 1. To obtain a function with specific curvature ?, we define f?R (X) = ?f (X) + (1 ? ?)—X— as in Equation (3). In all our experiments, we take the average over 20 random draws of R. We first set ? = 1 and vary . Figure 1(a) shows the empirical approximation factors obtained using EA and MUB, and the theoretical bound. The empirical factors follow the theoretical results very closely. Empirically, we also see that the problem becomes harder as decreases. Next we fix = 0.1 and vary the curvature ? in f?R . Figure 1(b) illustrates that the theoretical and empirical approximation factors improve significantly as ? decreases. Hence, much better approximations than the previous theoretical lower bounds are possible if ? is not too large. This observation can be very important in practice. Here, too, the empirical upper bounds follow the theoretical bounds very closely. Figures 1(c) and (d) show results for larger ? and ? = 1. In Figure 1(c), as ? increases, the empirical factors improve. In particular, as predicted by the theoretical bounds, EA outperforms MUB for large ? and, for ? ? n2/3 , EA finds the optimal solution. In addition, Figures 1(b) and (d) illustrate the theoretical and empirical effect of curvature: as n grows, the bounds saturate and approximate a constant 1/(1 ? ?) ? they do not grow

polynomially in n. Overall, we see that the empirical results quite closely follow our theoretical results, and that, as the theory suggests, curvature significantly affects the approximation factors.

6

Conclusion and Discussion

In this paper, we study the effect of curvature on the problems of approximating, learning and minimizing submodular functions under constraints. We prove tightened, curvature-dependent upper bounds with almost matching lower bounds. These results complement known results for submodular maximization [3, 29]. Given that the functional form and effect of the submodularity ratio proposed in [5] is similar to that of curvature, an interesting extension is the question of whether there is a single unifying quantity for both of these terms. Another open question is whether a quantity similar to curvature can be defined for subadditive functions, thus refining the results in [1] for learning subadditive functions. Finally it also seems that the techniques in this paper could be used to provide improved curvature-dependent regret bounds for constrained online submodular minimization [15]. Acknowledgments: Special thanks to Kai Wei for pointing out that Corollary 3.4 holds and for other discussions, to Bethany Herwaldt for reviewing an early draft of this manuscript, and to the anonymous reviewers. This material is based upon work supported by the National Science Foundation under Grant No. (IIS-1162606), a Google and a Microsoft award, and by the Intel Science and Technology Center for Pervasive Computing. Stefanie Jegelka?s work is supported by the Office of Naval Research under contract/grant number N00014-11-1-0688, and gifts from Amazon Web Services, Google, SAP, Blue Goji, Cisco, Clearstory Data, Cloudera, Ericsson, Facebook, General Electric, Hortonworks, Intel, Microsoft, NetApp, Oracle, Samsung, Splunk, VMware and Yahoo!. 8

2 References

[1] M. F. Balcan, F. Constantin, S. Iwata, and L. Wang. Learning valuation functions. COLT, 2011. [2] N. Balcan and N. Harvey. Submodular functions: Learnability, structure, and optimization. In Arxiv preprint, 2012. [3] M. Conforti and G. Cornuejols. Submodular set functions, matroids and the greedy algorithm: tight worstcase bounds and some generalizations of the Rado-Edmonds theorem. Discrete Applied Mathematics, 7(3): 251?274, 1984. [4] W. H. Cunningham. Decomposition of submodular functions. Combinatorica, 3(1):53?68, 1983. [5] A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In ICML, 2011. [6] G. Goel, C. Karande, P. Tripathi, and L. Wang. Approximability of combinatorial problems with multi-agent submodular cost functions. In FOCS, 2009. [7] G. Goel, P. Tripathi, and L. Wang. Combinatorial problems with discounted price functions in multi-agent systems. In FSTTCS, 2010. [8] M. Goemans, N. Harvey, S. Iwata, and V. Mirrokni. Approximating submodular functions everywhere. In SODA, pages 535?544, 2009. [9] R. Hassin, J. Monnot,

and D. Segev. Approximation algorithms and hardness results for labeled connectivity problems. J Combinatorial Optimization, 14(4):437?453, 2007. [10] S. Iwata and K. Nagano. Submodular function minimization under covering constraints. In In FOCS, pages 671?680. IEEE, 2009. [11] R. Iyer and J. Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. In UAI, 2012. [12] R. Iyer, S. Jegelka, and J. Bilmes. Curvature and Optimal Algorithms for Learning and Optimization of Submodular Functions: Extended arxiv version, 2013. [13] R. Iyer, S. Jegelka, and J. Bilmes. Fast semidifferential based submodular function optimization. In ICML, 2013. [14] S. Jegelka. Combinatorial Problems with submodular coupling in machine learning and computer vision. PhD thesis, ETH Zurich, 2012. [15] S. Jegelka and J. Bilmes. Online submodular minimization for combinatorial structures. ICML, 2011. [16] S. Jegelka and J. A. Bilmes. Approximation bounds for inference using cooperative cuts. In ICML, 2011. [17] S. Jegelka and J. A. Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In CVPR, 2011. [18] P. Kohli, A. Osokin, and S. Jegelka. A principled deep random field for image segmentation. In CVPR, 2013. [19] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In Proceedings of Uncertainity in Artificial Intelligence. UAI, 2005. [20] E. Lawler and C. Martel. Computing maximal ?polymatroidal? network flows. Mathematics of Operations Research, 7(3):334?347, 1982. [21] H. Lin and J. Bilmes. Optimal selection of limited vocabulary speech corpora. In Interspeech, 2011. [22] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In The 49th Meeting of the Assoc. for Comp. Ling. Human Lang. Technologies (ACL/HLT-2011), Portland, OR, June 2011. [23] H. Lin and J. Bilmes. Learning mixtures of submodular shells with application to document summarization. In UAI, 2012. [24] E. Nikolova. Approximation algorithms for offline risk-averse combinatorial optimization, 2010. [25] J. Soto and M. Goemans. Symmetric submodular function minimization under hereditary family constraints. arXiv:1007.2140, 2010. [26] P. Stobbe and A. Krause. Learning fourier sparse set functions. In International Conference on Artificial Intelligence and Statistics (AISTATS), 2012. [27] Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In FOCS, pages 697?706, 2008. [28] L. G. Valiant. A theory of the learnable. Communications of the ACM, 27(11):1134?1142, 1984. [29] J. Vondr?ak. Submodularity and curvature: the optimal algorithm. RIMS Kokyuroku Bessatsu, 23, 2010. [30] P.-J. Wan, G. Calinescu, X.-Y. Li, and O. Frieder. Minimum-energy broadcasting in static ad hoc wireless networks. Wireless Networks, 8:607?617, 2002. [31] P. Zhang, J.-Y. Cai, L.-Q. Tang, and W.-B. Zhao. Approximation and hardness results for label cut and related problems. Journal of Combinatorial Optimization, 21(2):192?208, 2011.

9