Monotone Submodular Maximization over a Matroid

Yuval Filmus

January 31, 2013

Abstract

In this talk, we survey some recent results on monotone submodular maximization over a matroid. The survey does not purport to be exhaustive.

1 Definitions

1.1 Submodularity

In the classical optimization problem of maximum coverage, we are given a collection of sets $S = \{S_1, \ldots, S_m\}$ and a number r. The goal is to select r sets whose union is as large as possible. The objective function f takes as input a subset X of S, and return the total number of elements covered by X. The function f enjoys several properties:

1. f is normalized: $f(\emptyset) = 0$.
2. f is monotone: if $A \subseteq B$ then $f(A) \leq f(B)$.
3. f is submodular: if $A \subseteq B$ and $S_i \notin B$ then $f(A + S_i) - f(A) \geq f(B + S_i) - f(B)$. (This property is also known as *diminishing returns*.)

To see that the last property holds, note that the *marginal* $f(A + S_i) - f(A)$ counts the number of elements in $S_i \setminus \bigcup A$. Since $A \subseteq B$, this is not smaller than the number of elements in $S_i \setminus \bigcup B$.

Here is another example: maximum directed cut. We are given a directed graph $G = (V, E)$. The goal is to select a subset S of the vertices that maximizes the number of edges between S and $V \setminus S$. This time the objective function f takes as input a subset S of the vertices, and returns the number of edges between S and $V \setminus S$. The function f enjoys the following properties:

1. f is normalized: $f(\emptyset) = 0$.
2. f is non-negative: $f(S) \geq 0$.
3. f is submodular: if $A \subseteq B$ and $x \notin B$ then $f(A + x) - f(A) \geq f(B + x) - f(B)$.

To see that the last property holds, note that $f(A + x) - f(A)$ is the difference between the number of edges from x to $S \setminus (A + x)$ and the number of edges from A to x. Going from A to B, the first number decreases while the second increases.

The general setting is that of a *set function* $f \colon 2^U \to \mathbb{R}$, where U is some finite universe. All set functions we consider in this talk will be submodular and non-negative. In many applications the function in question is also monotone and normalized, and in this case better approximation algorithms are possible.

Another possibility is that the function is *symmetric*, that is it satisfies $f(S) = f(U \setminus S)$. The prototypical example is maximum cut, which is maximum directed cut for an undirected graph. (Maximum cut is actually more well-known than the more general maximum directed cut.)

Submodularity is usually formulated in an equivalent form:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B).$$

We leave the reader to show that this is equivalent to the condition considered above.

Throughout this talk we only consider *maximization* of submodular functions. There are several different algorithms for *minimizing* submodular functions. These algorithms, none of which is simple, are able to find the optimum in polynomial time. In this sense, submodular functions are like convex functions, for which minimization is much easier than maximization. (On the other hand, the multilinear relaxation discussed below is actually concave.)

1.2 Matroids

In the case of maximum directed (or undirected) cut, the optimization problem we are interested in is unconstrained. However, this does not make sense for the monotone case, since then the answer is always U. Instead, optimization is done over a collection of *feasible sets*. For maximum coverage, this collection consists of all sets of size r. We call this type of constraint a *uniform matroid* of rank r.

A more complicated situation arises in the problem of Max-SAT. In this problem, we are given a list of clauses, and the goal is to find an assignment which satisfies as many of the clauses as possible. We can represent each value of each variable by the set of clauses it satisfies, and then the objective function becomes a set coverage function. The collection of feasible sets consists of all assignments of values to variables.

Generalizing, suppose that the universe U is partitioned into r parts $P_1, \ldots, P_r$ (in Max-SAT, each part consists of the two possible assignments to each variable). A set is feasible if it contains exactly one point out of each part. This type of constraint is called a *partition matroid* of rank r. If all parts contain the same elements (same in function, but different in name) then this is the same as optimization over a uniform matroid. In other words, partition matroids generalize uniform matroids.

Our algorithms will work for even more general types of constraints called *matroids*. In fact, all algorithms for partition matroids can be adapted to general matroids. Some relevant nomenclature is *bases* (the same as our feasible sets) and *independent sets* (subsets of our feasible sets).

2 Algorithms

2.1 Monotone submodular maximization over uniform matroids

The classical problem of maximum coverage is a special case of monotone submodular maximization over a uniform matroid. The greedy algorithm, which is optimal for maximal coverage, is as effective in the more general case.

Let the monotone submodular function in question be f, and let the uniform matroid have rank r. The greedy algorithm proceeds in r phases. In phase i, given the already chosen points $S_1, \ldots, S_{i-1}$, we choose $S_i \in U$ as the point maximizing $f(\{S_1, \ldots, S_{i-1}\})$. It can be shown that this algorithm achieves an approximation ratio of $1 - 1/e$. (In fact, for given r the optimal approximation guarantee is $1 - (1 - 1/r)^r > 1 - 1/e$.)

Here is a tight set coverage instance. The universe is $U = \{1, \ldots, r\}^r$. Let S_i be the set containing all elements whose ith coordinate is 1, and let O_i be the set containing all elements whose coordinates sum to a value of the form $ar + i$. The sets S_i together cover $1 - (1 - 1/r)^r$ of the universe, and one can show that they are legitimate choices of the greedy algorithm. In contrast, the sets O_i cover the entire universe.

2.2 Monotone submodular maximization over general matroids

Maximum coverage becomes much more interesting when we replace the uniform matroid constraint by a partition matroid constraint. The performance of the greedy algorithm drops from $1 - 1/e$ to $1/2$, as the following example shows. One partition consists of the sets $\{x, \epsilon\}, \{y\}$, and the other consists of the set $\{x\}$.

Here ϵ is an element of small weight. The greedy algorithm chooses the set $\{x, \epsilon\}$ from the first partition, and is then forced to choose the set $\{x\}$ from the other partition, covering elements of total weight $1 + \epsilon$, compared to the optimum 2.

2.2.1 The continuous greedy algorithm

In a breakthrough result, Calinescu, Chekuri, Pál and Vondrák [2] were able to utilize the *multilinear relaxation* to come up with a $1 - 1/e$ approximation algorithm. The multilinear relaxation associated with a function f is a function $F\colon [0,1]^U \to \mathbb{R}$ defined as follows. Denote the input corresponding to an element $u \in U$ by p_U. Let S be a random set which contains each $u \in U$ with probability p_u, independently. The value of F is the expectation of $f(S)$. Note that in general, we cannot evaluate F in polynomial time, but we can get a good approximation by sampling. If the input is 0/1, then the two functions have identical values, and that's why F is called a relaxation of f. It turns out that since f is monotone, $\partial F/\partial p_u \geq 0$, and since f is submodular, $\partial^2 F/\partial p_u \partial p_v \leq 0$, and so F is concave.

The continuous greedy algorithm attempts to find an optimum of F over the *matroid polytope*, which in the case of partition matroids has the following simple description. It consists of the set of points $(p_1, \ldots, p_{|U|})$ such that for each part P, $\sum_{u \in P} p_u = 1$. In other words, for each part P we have a probability distribution over the possible points. The concavity of F implies that the maximum is attained at a vertex (a 0/1 point), and so the maximum of F over the matroid polytope is equal to the maximum of f over the matroid. Conversely, given an approximate optimum of F, we can round it to an approximate optimum of f by picking one element from each partition according to the probability distribution of each part. It turns out that if we do that, the expected value of f over the set is lower-bounded by the value of F.

The continuous greedy algorithm generates a sequence of points $x_0, x_\epsilon, x_{2\epsilon}, \ldots, x_1$, where $\epsilon = 1/N$ is a small enough number. Each point $x_{t\epsilon}$ satisfies $\sum x_{t\epsilon} = t\epsilon$, so it belongs to the matroid polytope scaled by $t\epsilon$. The starting point is the zero vector x_0. Given $x_{t\epsilon}$, we want to choose a vector $x_{(t+1)\epsilon}$ that maximizes the value of F. Let $x_{(t+1)\epsilon} = x_{t\epsilon} + \epsilon y$, where y belongs to the matroid polytope. We will use the approximation

$$F(x_{(t+1)\epsilon}) \approx F(x_{t\epsilon}) + \epsilon \langle \nabla F(x_{t\epsilon}), y \rangle.$$

We can estimate $\nabla F(x_{t\epsilon})$ by sampling, using the formula

$$\frac{\partial F}{\partial x_i} = F(x|_{i=1}) - F(x|_{i=0}).$$

The optimal y is then obtained by choosing the coordinate maximizing $\nabla F(x_{t\epsilon})$ in each part.

Amazingly, it turns out that $F(x_1)$ is roughly a $1 - 1/e$ approximation for the optimum of F. As $\epsilon \to 0$, the function $\phi(t\epsilon) = F(x_{t\epsilon})$ satisfies $\phi'(t) \geq f(O) - \phi(t)$, where O is an optimum; we get this inequality by considering the choice $y = 1_O$. It turns out that any function satisfying this differential inequality is lower-bounded by the solution to the corresponding differential equation, which is $(1 - e^{-t})f(O)$.

2.2.2 The non-oblivious local search algorithm

An alternative algorithm was devised by Filmus and Ward [7], using the paradigm of non-oblivious local search. The idea is to define a related function g and find a *local optimum* of g, using the local search algorithm. This algorithm starts with S being the result of running the greedy algorithm on g and the matroid. We then repeatedly try to exchange one point of S for another in a way which increases the value of g while keeping the set feasible (in the matroid). Eventually, the process will converge to a local optimum S, which is a set with the property that exchanging one point with another never results in a feasible set having a larger value of g. It turns out that with the correct choice of g, a local optimum of g is also a $1 - 1/e$ approximation of f!

When f is a coverage function, the function g is obtained by giving more weight to elements appearing more than once in the sets in the input. In the general case, g is of the form $g(S) = \sum_{T \subseteq S} c(|S|, |T|) f(T)$

for some constants $c(s,t)$ depending only on the sizes of the sets involved:

$$c(s,t) = \frac{1}{e-1}\int_0^1 p^{t-1}(1-p)^{s-t}e^p \,\mathrm{d}p.$$

Again, we see that g cannot be evaluated exactly in polynomial time, but it can be approximated by sampling.

The idea of the proof is to consider a local optimum $S = \{S_1, \ldots, S_r\}$ and a global optimum $O = \{O_1, \ldots, O_r\}$. Local optimality implies $g(S) \geq g(S - S_i + O_i)$, and we prove that

$$\sum_{i=1}^{r}[g(S) - g(S - S_i + O_i)] \geq 0 \Longrightarrow f(S) \geq \left(1 - \frac{1}{e}\right) f(O).$$

This boils down to taking a positive linear combination of the inequalities stating that f is monotone, submodular and normalized.

2.3 Unconstrained non-monotone submodular maximization

In the non-monotone case, optimal results have so far been obtained only for unconstrained maximization. The problem is especially easy if the submodular function is symmetric: in that case a random subset of U is always a 1/2-approximation [5]. This is a consequence of the identity

$$\mathbb{E}[f(R)] \geq \frac{1}{4}f(U) + \frac{1}{4}f(O) + \frac{1}{4}f(U \setminus O) + \frac{1}{4}f(\emptyset),$$

where R is a uniformly random set. (Slightly better results are obtained by choosing R to be of size $|U|/2$.)

A (surprisingly simple) optimal algorithm for the non-symmetric case was found by Buchbinder, Feldman, Naor and Schwartz in 2012 [1]. The algorithm outputs a vector $\mathbf{p}$ satisfying $F(\mathbf{p}) \geq F(\mathbf{q})/2$, where F is the multilinear relaxation and $\mathbf{q}$ is a maximum of F. The idea is to list the elements of U in sequence: $U = \{u_1, \ldots, u_n\}$, and decide on each of them in turn. Suppose we have decided on values for $p_1, \ldots, p_{t-1}$. To find the value of p_{t+1}, let

$$a_t = F(p_1, \ldots, p_{t-1}, 1, 0, \ldots, 0) - F(p_1, \ldots, p_{t-1}, 0, 0, \ldots, 0),$$
$$b_t = F(p_1, \ldots, p_{t-1}, 0, 1, \ldots, 1) - F(p_1, \ldots, p_{t-1}, 1, 1, \ldots, 1).$$

Also, let

$$x_t = F(p_1, \ldots, p_{t-1}, 0, \ldots, 0),$$
$$y_t = F(p_1, \ldots, p_{t-1}, 1, \ldots, 1),$$
$$z_t = F(p_1, \ldots, p_{t-1}, q_t, \ldots, q_n).$$

It turns out that $a_t + b_t \geq 0$ always. If $a_t \leq 0$ then we put $p_t = 0$. If $b_t \leq 0$ then we put $p_t = 1$. Otherwise, $p_t = a_t/(a_t + b_t)$. In effect, at time t we're considering two possible solutions, x_t and y_t.. The values a_t, b_t are used to find a value of p_t which is "favorable" with regard to $F(x_{t+1}), F(y_{t+1})$. The idea of the proof is showing that

$$F(z_{t+1}) + \frac{F(x_{t+1}) + F(y_{t+1})}{2} \geq F(z_t) + \frac{F(x_t) + F(y_t)}{2}.$$

Since $z_1 = q$ and $z_{t+1} = x_{t+1} = y_{t+1} = p$, this implies that $2F(p) \geq F(q)$.

Since we're evaluating F at general points, this algorithm is randomized. There is a variant of the algorithm which is deterministic. In this variant, in the case where both a_t and b_t are positive, we put $p_t = 1$ if $a_t \geq b_t$ and $p_t = 0$ if $a_t < b_t$. One can show that

$$F(z_{t+1}) + F(x_{t+1}) + F(y_{t+1}) \geq F(z_t) + F(x_t) + F(y_t),$$

and so $3F(q) \geq F(p)$. The result is a deterministic 1/3 approximation algorithm running in linear time.

The algorithm exploits the fact that submodular functions are oblivious to the distinction between 0s and 1s. This is because if f is a submodular function then so is the function f' given by $f(S) = f(U \setminus S)$. This symmetry is of course lost in the monotone case.

3 Hardness

Previously, we have claimed that various algorithm are optimal. How do we know that, and what does it even mean for an algorithm to be optimal? In the simplest case, monotone submodular maximization over a uniform matroid, the answer is particularly satisfying. Feige [4] has shown using PCP techniques that if there is an algorithm that approximates maximum coverage better than $1 - 1/e$ then P=NP. This is one sense in which $1 - 1/e$ is optimal.

There is another sense in which $1-1/e$ is optimal. Suppose that the submodular function is given to us as a black box, known in this case as a *value oracle*. Nemhauser and Wolsey [8] showed that an approximation algorithm whose performance is better than $1 - 1/e$ must make exponentially many queries to the value oracle. The idea of the proof is to come up with two functions f_1, f_2 which look the same for almost all inputs, but whose respective optimal values differ by a factor of $1 - 1/e$.

In the case of unconstrained non-monotone submodular maximization, there is no optimal NP-hardness result, but Feige, Mirrokni and Vondrák [5] proved that in the value oracle setting, any algorithm whose performance is better than $1/2$ must make exponentially many queries. The idea of the proof is the same, only this time the optimal values differ by the larger factor of $1/2$.

More concretely, $f_1(S) = |S|(n - |S|)$ (here $n = |U|$). To define f_2, pick a random partition $C \cup D$ of U. On inputs S where $|S \cap C| \approx |S \cap D|$, $f_2(S) = f_1(S)$. On inputs where the two quantities $c = |S \cap C|$ and $d = |S \cap D|$ are far apart, $f_2(S) \approx 2c(n/2 - d) + 2d(n/2 - c)$. The function f_1 is just the cut function of a complete graph, and its optimum is $(n/2)^2$. When c and d are far apart, the function f_2 is close to the cut function of the complete bipartite graph on (C, D) with double edges, whose optimum is $2(n/2)^2 = n^2/2$. A randomized approximation algorithm doesn't know the partition (C, D), so with high probability it will never query the function at a point in which c and d are far apart. Therefore it cannot distinguish between f_1 and f_2.

The difficult part of the proof is constructing a function f_2 satisfying the properties above while being submodular. Vondrák [9] has come up with a general way of doing that. The lower bound that his method achieves depends on the *symmetry gap*, which is the gap between a symmetric solution and an arbitrary solution for a simple base instance, using which the actual hard-to-distinguish instances are constructed.

Dobzinski and Vondrák [3] were able to strengthen the value oracle results to hardness results on functions given explicitly, conditional on the assumption NP≠RP. The idea is to use a reduction from Unique-SAT (that's why they get RP instead of P). Given a formula ϕ which is either unsatisfiable or has a unique satisfying assignment, we can write an explicit function f that looks like f_1 if ϕ is unsatisfiable and like f_2 if ϕ is satisfiable. The partition (C, D) corresponds to the satisfying assignment α of ϕ. The trick is that we only need to determine the partition (C, D) if the input S is very skewed with respect to the partition. The encoding of α is such that given a skewed input S, we can recover α and so (C, D) using list decoding. List decoding can handle larger error rates, and in this case it is easy to find the correct satisfying assignment from the list produced by the decoding algorithm by checking that it satisfies ϕ.

4 Open questions

Here is a list of some of the more important open questions in this area.

- Optimization over multiple matroids: The most important open question is finding the optimal approximation ratio for more general domains of feasible sets, the most important of which is the intersection of multiple matroids. An example of the latter is maximizing over the set of maximum matchings in a complete bipartite graph.
- Constrained non-monotone submodular maximization: Given a non-monotone submodular function and a matroid, two possible domains of feasible sets are the independent sets and the bases of the matroid. Both cases have been considered, but the optimal approximation ratio is still unknown.
- Randomized versus deterministic algorithms: Both algorithms for monotone submodular maximization over a matroid are randomized. The best-known deterministic algorithm is the greedy algorithm,

which only gives a $1/2$ approximation, compared to the $1 - 1/e$ approximation given by the former algorithms (there does exist an optimal deterministic $1 - 1/e$ algorithm for maximum coverage over a matroid [6]). In the unconstrained non-monotone case, the best known deterministic algorithm gives a $1/3$ approximation, but the optimal randomized approximation ratio is $1/2$. Could it be that deterministic algorithms can achieve less than randomized algorithms in this case, in the value oracle setting? This happens in other settings, for example algorithms finding a zero of a balanced Boolean set function: a deterministic algorithm requires $2^{|U|-1}$ queries, while a randomized algorithm requires k queries to succeed with probability $1 - 2^{-k-1}$.

References

[1] Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *FOCS*, 2012.

[2] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint. *SICOMP*, 40(6):1740–1766, 2011.

[3] Shahar Dobzinski and Jan Vondrák. From query complexity to computational complexity. In *STOC*, pages 1107–1116, 2012.

[4] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, July 1998.

[5] Uriel Feige, Vahab Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. *SICOMP*, 40:1133–1153, 2011.

[6] Yuval Filmus and Justin Ward. The power of local search: Maximum coverage over a matroid. In *STACS*, pages 601–612, 2012.

[7] Yuval Filmus and Justin Ward. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. *arxiv.org*, 1204.4526, 2012.

[8] G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Research*, 3(3):177–188, 1978.

[9] Jan Vondrák. Symmetry and approximability of submodular maximization problems. *arxiv.org*, 1110.4860, 2011.