lions@epfl

EE-731: ADVANCED TOPICS IN DATA SCIENCES
LABORATORY FOR INFORMATION AND INFERENCE SYSTEMS
SPRING 2016

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

INSTRUCTOR: VOLKAN CEVHER

SCRIBER: ABBAS BAZZI AND ASHKAN NOROUZI-FARD

SUBMODULAR OPTIMIZATION

Many interesting problems in computer science can be casted as optimization problems, where given a function f, one would want to find either the minimizer or the maximizer of f subject to some constraints. An interesting class of such functions are set functions of the form $f : \{0,1\}^V \mapsto \mathbb{R}$, i.e., we are given a ground set V, and for every subset $S \subseteq V$, the function f assigns a value. The goal in such problems is find a *feasible* subset $S^\star$ such that $f(S^\star)$ is maximized (or minimized). The general version of this problem is hard, and cover a large fraction of the interesting problems in computer science. Hence in this lecture we only focus on optimizing over such set functions when they are either submodular, supermodular, or modular. We then study how to optimize over more general functions when the underlying constraints satisfy additional property.

1 Submodular Optimization

In this lecture, we will mainly focus on the following three topics:

- Submodular Function Maximization
- Submodular Function Minimization
- Interger programming

Let us start by recalling the definition of submodular functions.

Definition 1. *A function $f : 2^V \to \mathbb{R}$ on a ground set V is said to be submodular if for all $S \subseteq T \subseteq V$ and any $e \in V \backslash T$, it holds $\Delta(e|S) \geq \Delta(e|T)$, where $\Delta(e|S) = f(S \cup \{e\}) - f(S)$.*

We also call f monotone, if for all $S \subseteq T$, we have that $f(S) \leq f(T)$.

In several problems in theoretical computer science, machine learning and game theory, the goal is to optimize over a submodular function, i.e., finding the minimum or the maximum of such function. Hence we define the following two problems.

Definition 2. SFMin: *Given a submodular function $f : 2^V \to \mathbb{R}$ the goal is to find a set S that minimizes $f(S)$.*

Definition 3. SFMax: *Given a submodular function $f : 2^V \to \mathbb{R}$ the goal is to find a set S that maximizes $f(S)$.*

We first focus on the SFMax problem, and show that not only it is *NP*-hard, but it is also unconditionally hard. We also present an $(1 - \frac{1}{e})$-approximation algorithm for a special case of this problem. Then we focus on SFMin, and show that, unlike SFMax, it is solvable in polynomial.

2 Submodular Maximization

We start by studying submodular functions in this section. We start in Section 2.1 by motivating the use of sub modular function with application in Learning-based CS. We then show in Section 2.2 that SFMax is NP-hard, and distinguish between two different computational models for this problem. Then we study modular maximisation in Section 2.3, and show that this problem can be solved exactly in polynomial time, unlike submodular maximization. For the latter, we design in Section 2.4 a $(1 - \frac{1}{e})$-approximation algorithm when we require our submodular function to be monotone, and we add cardinality constraints for a feasible solutions. We then show in Section 2.5 how to use a greedy algorithm to solve the Minimum submodular set cover problem.

2.1 Submodular maximization in Learning-based CS

In this section we present three examples of submodular maximization in learning based CS.

Definition 4. *Given a set of m training signals $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \mathbb{C}^p$, find an index set Ω of a given cardinality n such that a related test signal $\boldsymbol{x}$ can reliably be recovered given the subsampled measurement vector $\mathbf{b} = \P_\Omega \Psi \boldsymbol{x}$.*

Definition 5. Average energy criterion

$$\hat{\Omega} = \arg\max_{\Omega\,:\,|\Omega|=n} \frac{1}{m}\sum_{j=1}^{m}\sum_{i\in\Omega}|\langle\psi_i, \boldsymbol{x}_j\rangle|^2$$

This is a cardinality constrained modular maximization problem.

Definition 6. Generalized average energy criterion

$$\hat{\Omega} = \arg\max_{\Omega\,:\,|\Omega|=n} \frac{1}{m}\sum_{j=1}^{m} g\Big(\sum_{i\in\Omega}|\langle\psi_i, \boldsymbol{x}_j\rangle|^2\Big).$$

where $g : \mathbb{R} \to \mathbb{R}$ be an increasing concave function with $g(0) = 0$. This is a cardinality constrained monotone submodular maximization problem.

2.2 SFMax is *NP*-hard

Let us first show that this problem is *NP*-hard. In order to do that, we first define our computational model, as it is different than usual models. To see the difference, assume that we are given as input the value of $f(S)$ for any $S \subseteq V$, then we can just check all of these value, and find min/max in linear time (in the size of the input). However, this renders the size of the input exponentially large compared to the size of the ground set, and any algorithm (even the linear time) would be extremely slow. Therefore, we normally assume that we are only given the ground set in the input, and we have a polynomial size description of a Turing machine that allows us to query the value of $f(S)$ for any $S \subseteq V$. In particular, this turing machine returns the value of any query in polynomial time with respect to the size of the ground set.

In order to show that SFMax is NP-hard, reduce form SFMax to Max Cut. To do that, we start by defining the Max Cut problem and then show that the *cut function* is submodular.

Definition 7. Max Cut: *Given a graph $G = (V, E)$, find a subset of vertices $S \subseteq V$ maximizing the number of edges between S and $V\backslash S$.*

$$max_{S\subseteq V}|\delta(S)|$$

Max Cut is known to be an *NP*-hard problem, and is listed in Karp's list of 21 NP-complete problems [4]. The following Lemma shows that the cut function is submodular.

Lemma 2.1. *For a given graph $G = (V, E)$, $\delta(S)$ is a submodular function, where $\delta(S) = \{(u, v) \in E : u \in S, v \in V\backslash S\}$*

Proof. Let $S \subseteq T \subseteq V$ and $e \in V\backslash T$. We need to show that $\Delta(e|T) \leq \Delta(e|S)$, where $\Delta(e|A) = \delta(A \cup u) - \delta(A)$. Let us define the following three variables:

- n_1 be the number edges between e and $V\backslash T$.
- n_2 be the number edges between e and $T\backslash S$.
- n_3 be the number edges between e and S.

Now one can easily see that $\Delta(e|S) = n_1 + n_2 - n_3$ and $\Delta(e|T) = n_1 - n_2 - n_3$, since $n_2 \geq 0$, we get that $\Delta(e|T) \leq \Delta(e|S)$. □

This shows that if SFMax is solvable in polynomial time, then Max Cut is solvable in polynomial time as well, hence SFMax in *NP*-hard. Note that its easy to design a Turing machine that can find the number of the edges in a cut.

In a different computational model, we also can show that any $(1/2 + \epsilon)$-approximation algorithm requires exponentially many oracle calls. This means that it is *impossible* to design a polynomial time algorithm that gives an approximation guarantee that is strictly better than 1/2 with fewer oracle requests [3]. Note that, this result *does not* show that approximating SFMax within a factor 1/2 is NP-hard, but in a separate result we get that it is indeed *NP*-hard to approximate SFMax within a factor strictly better that 1/2. There is also a polynomial 1/2-approximation algorithm for SFMax.

2.3 Modular Function Maximization

We focus in this section on Modular function maximisation; while SFMax is hard, MFMax (Modular Function Maximization) is easy.

Definition 8. Unconstrained modular maximization *Given a ground set V and constants $c_1, ..., c_n \in \mathbb{R}$, the goal is to find a set $S \subseteq V$ maximizing $f(S)$, where $f(S) := \sum_{i\in S} c_i$.*

$$max_{S\subseteq V} f(S)$$

For solving this problem we just need to pick elements with positive c_i value, which is trivially linear time.

Definition 9. Cardinality Constrained modular maximization *Given a ground set V and constants $c_1, ..., c_n \in \mathbb{R}$, the goal is to find a set $S \subseteq V$ such that $|S| \leq k$, maximizing $f(S)$, where $f(S) := \sum_{i\in S} c_i$.*

$$max_{|S|\leq k} f(S)$$

For solving this we just pick the elements with k biggest c_i as while as they are positive. Similar to previous part this can still be done in linear time. That is, we can find the k-th biggest element of a set in time $O(n)$.

2.4 Cardinality constrained monotone submodular maximization

In this section we present a simple but tight algorithm for cardinality constrained monotone submodular maximization problem. In this problem, we are given a ground set V and an oracle, the goal is to find a set S, such that $|S| \leq k$, maximizing $f(S)$, where f is a monotone, submodular function.
Here is the greedy algorithm:

1. Initialize $S_0 = \emptyset$
2. For $i = 1, \ldots, k$
   - Find $e_i = \arg\max_{e \in V \setminus S_{i-1}} \Delta(e|S_{i-1})$
   - Set $S_i = S_{i-1} \cup \{e_i\}$
3. Return S_k

The following lemma shows that the greedy algorithm gives us a $(1 - 1/e)$-approximation factor.

Lemma 2.2. *For any monotone submodular function f, with $f(\emptyset) = 0$, after ℓ iterations of the greedy algorithm, it holds that*

$$f(S_\ell) \geq (1 - e^{-\ell/k}) f(S^\star),$$

where $S^\star$ is the optimal solution.

Proof. We can assume w.l.o.g. that $|S^\star| = k$, since by monotonicity of f adding elements can only increase the function value. Assume that $S^\star = \{e_1^\star, e_2^\star, \cdots, e_k^\star\}$, then for all $i \leq \ell$,

$$\begin{aligned}
f(S^\star) &\leq f(S^\star \cup S_i) \\
&= f(S_i \cup \{e_1^\star, e_2^\star, \cdots, e_{k-1}^\star\}) + \Delta\left(e_k^\star | S_i \cup \{e_1^\star, e_2^\star, \cdots, e_{k-1}^\star\}\right) \\
&= f(S_i) + \sum_{j=1}^{k} \Delta\left(e_j^\star | S_i \cup \{e_1^\star, e_2^\star, \cdots, e_{j-1}^\star\}\right) \\
&\leq f(S_i) + \sum_{j=1}^{k} \Delta\left(e_j^\star | S_i\right) \\
&\leq f(S_i) + \sum_{j=1}^{k} \Delta\left(e_i | S_i\right) \\
&\leq f(S_i) + k\left(f(S_{i+1}) - f(S_i)\right)
\end{aligned}$$

The first inequality holds by monotonicity, the third one follows by repeating the second inequality k times. The forth one follows by submodularity.

Let $\Delta f_i := f(S^*) - f(S_i)$. Then by rearranging the term we get that, $\Delta f_{i+1} \leq \left(1 - \frac{1}{k}\right) \Delta f_i$.

$$f(S^\star) - f(S_\ell) \leq (1 - \frac{1}{k})^\ell f(S^\star) \leq e^{-l/k} f(S^*)$$

Rearranging terms yields $f(S_l) \geq (1 - e^{-l/k}) f(S^*)$.

□

After k iterations, the greedy algorithm achieves a $(1 - 1/e)$-approximation, which is tight; no algorithm requiring a *polynomial* number of oracle calls has a better performance. This results also shows something more interesting is shows that we can get much better guarantee if we are allowed to choose a set of size more than k and still compare our solution with a solution of size k. For instance we get that by choosing $l = 5k$, $f(S_{5k}) \geq 0.9933 \max_{|S| \leq k} f(S)$.

In the next section, we use the greedy algorithm to solve Minimum submodular set cover problem.

2.5 Minimum Submodular Set Cover

We first define the problem, and then show that the greedy algorithm can almost solve this problem.

Definition 10. *Given a fixed $z \in Z$, such that $z \leq f(V)$, if we now run the greedy algorithm until we reach $f(S_\ell) \geq z$, how big is the size of S_l, compared to the minimum sized $S^\star$ set achieving $f(S^\star) \geq z$.*

$$k^\star = \min_{S \subseteq V} \{|S| : f(S) \geq z\}$$

The following theorem shows that the $|S_\ell|$ that the greedy algorithm finds is not that bad. Recall that a function f is called normalized, if $f(\emptyset) = 0$.

Lemma 2.3. *Given a normalized monotone submodular integer-valued function f and a fixed $z \in Z$, such that $z \leq f(V)$. Let ℓ be smallest integer such that $f(S_\ell) \geq z$. Then*

$$\ell \leq \left(1 + \ln \max_{v \in V} f(\{v\})\right) k^\star$$

I

After showing that SFMax is a hard problem, and also presenting a tight approximation, it is time to focus on SFMin problem Before we do that, we end this section with an exercise.

Exercise 1. *Let $f : \{0,1\}^V \mapsto \mathbb{Z}$ be a submodular function such that $\max_{v \in V} f(v) \leq 1$. Is f modular?*

Solution. No. As a counter example, let $V = \{1, 2\}$, and define f as follows:

$$\begin{aligned} f(\emptyset) &= 0 \\ f(\{1\}) &= 1 \\ f(\{2\}) &= 1 \\ f(\{1,2\}) &= 1 \end{aligned}$$

f is clearly submodular, however it is not modular. ◂

3 Unconstrained Submodular Minimization

In this section we abuse the notation a little bit for the sake of readability. As defined before, a set function is defined as $f : 2^V \to \mathbb{R}$. In this section we also define it over a vector of size $n = |V|$, i.e., $f(\mathbb{1}_S) = f(S)$.

We start by defining the *Convex Closure*, and then in Section 3.1 we define the *Lovasz extention* and use them to give a polynomial time algorithm for SFMin in Section 3.2.

Definition 11. *Given any set function $f : \{0,1\}^n \to \mathbb{R}$, for all $x \in [0,1]^n$ we define the convex closure of f as:*

$$f^-(x) = \min_{\alpha \in [0,1]^{2^n}} \left\{ \sum_{S \subseteq V} \alpha_S f(S) : x = \sum_{S \subseteq V} \alpha_S \mathbb{1}_S, \sum_{S \subseteq V} \alpha_S = 1, \alpha_S \geq 0 \right\}$$

3.1 Lovász Extension

As we saw in the previous section, given a function $f : \{0,1\}^n \mapsto \mathbb{R}$, we wanted to define the convex closure of f, denoted by $f^- : [0,1]^n \mapsto \mathbb{R}$ that is *close* to f yet can be minimized efficiently. A good candidate for that purpose if the Lovász extension defined as follows:

Definition 12 (Lovász extension [6])**.** *Given a normalized set function $f : \{0,1\}^n \to \mathbb{R}$ (i.e., $f(\emptyset) = 0$), its Lovász extension $f_L : [0,1]^n \to \mathbb{R}$ is defined $\forall s \in [0,1]^n$ as follows:*

$$f_L(s) = \sum_{k=1}^{n} s_{j_k} \left(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\}) \right)$$

where $s_{j_1} \geq s_{j_2} \geq \cdots \geq s_{j_n}$.

The convex closure $f^-(x) = f_L(x)$ where f_L is the Lovász extension of f iff f is a submodular function [11]. We present below some observations about f_L and verify some of them.

Fact 1. *f_L is an extension of f since $f_L(s) = f(s), \forall s \in \{0,1\}^n$.*

Proof. For any $s \in \{0,1\}^n$ that is an indicator vectors of a set S, let $j_1, \cdots, j_n$ be such that $s_{j_1} \geq s_{j_2} \geq \cdots \geq s_{j_n}$, and define $\ell \in [n]$ to be the largest index j_k such that $s_{j_k} = 1$. It is easy to see that $S = \{j_1, j_2, \ldots, j_\ell\}$ hence from Definition 12 we get

$$\begin{aligned} f_L(s) &= \sum_{k=1}^{n} s_{j_k}(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})) \\ &= \sum_{k=1}^{\ell} (f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})) \\ &= f(\{j_1, \cdots, j_\ell\}) \\ &= f(S) \end{aligned}$$

where the third equality follows from the telescoping nature of the sum. □

Fact 2. *f_L can be computed efficiently in $O(n \log n)$.*

Proof. It is to see that given an ordering $s_{j_1} \geq s_{j_2} \geq \cdots \geq s_{j_n}$, $f_L(s)$ can be computed using $O(n)$ oracle accesses. Hence there running time of the computation is dominated by sorting the $s_j{}'$, which requires $O(n \log n)$ □

Fact 3. *For a fixed ordering of s, f_L is a linear function.*

Proof. Fix any $0 \leq a, b \leq 1$, and $s_1, s_2 \in [0,1]^n$ such that $s = as_1 + bs_2 \in [0,1]^n$, and consider a fixed ordering $s_{j_1}, \cdots, s_{j_n}$, then

$$\begin{aligned} f_L(s) &= \sum_{k=1}^{n} s_{j_k}(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})) \\ &= \sum_{k=1}^{n} (as_{1,j_k} + bs_{2,j_k})(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})) \\ &= a\sum_{k=1}^{n} s_{1,j_k}(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})) + b\sum_{k=1}^{n} s_{2,j_k}(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})) \\ &= af_L(s_1) + bf_L(s_2) \end{aligned}$$

where to get the last equality we used the fact that the ordering is fixed. □

Fact 4. *For modular functions, f_L is a linear function.*

Proof. Assume f is a modular function, then we get that for any $s \in [0,1]^n$

$$\begin{aligned} f_L(s) &= \sum_{k=1}^{n} s_{j_k}(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})) \\ &= \sum_{k=1}^{n} s_{j_k} f(\{j_k\}) \end{aligned}$$

hence f_L is linear. □

Fact 5. *Let $h = f + g$, then $h_L = f_L + g_L$.*

Proof. From Definition 12, we that for any $s \in [0,1]^n$:

$$\begin{aligned} h_L(s) &= \sum_{k=1}^{n} s_{j_k}\left(h(\{j_1, \cdots, j_k\}) - h(\{j_1, \cdots, j_{k-1}\})\right) \\ &= \sum_{k=1}^{n} s_{j_k}\left([f(\{j_1, \cdots, j_k\}) + g(\{j_1, \cdots, j_k\})] - [f(\{j_1, \cdots, j_{k-1}\}) + g(\{j_1, \cdots, j_{k-1}\})]\right) \\ &= \sum_{k=1}^{n} s_{j_k}\left(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})\right) + \sum_{k=1}^{n} s_{j_k}\left(g(\{j_1, \cdots, j_k\}) - g(\{j_1, \cdots, j_{k-1}\})\right) \\ &= f_L(s) + g_L(s) \end{aligned}$$

□

Fact 6. *f_L is positively homogenous, i.e., $f_L(\alpha s) = \alpha f_L(s), \forall \alpha > 0$.*

Fact 7. *f_L is a non-decreasing function if f is monotone.*

We note here that there are several definitions for the Lovász extension, however we stick to Definition 12 for our purposes.

The motivation behind defining the Lovász extension f_L was to come up with a function that is close to our submodular function f, and can be efficiently minimized. We have shown earlier that f_L is indeed close to f, hence it remains to show that the second property holds. To do that, we show that f_L is convex if and only if f is submodular, which yields the required property.

Theorem 3.1 ([6])**.** *Given a set function f and its Lovász extension f_L, f_L is convex iff f is submodular.*

Proof. We first show that f_L is convex if f is submodular. To see this, consider any $s_1, s_2 \in [0,1]^n$ and a real $0 \leq \lambda \leq 1$, and let $s = \lambda s_1 + (1-\lambda)s_2 \in [0,1]^n$ hence

$$\begin{aligned} f_L(s) &= \sum_{k=1}^{n} s_{j_k}(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})) \\ &= \lambda \sum_{k=1}^{n} s_{1,j_k}(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})) + (1-\lambda) \sum_{k=1}^{n} s_{2,j_k}(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})) \end{aligned} \tag{1}$$

$$\tag{2}$$

Now consider a variant of the Lovász extension where the ordering is fixed, i.e., $f_L^J(x)$ where I is sequence defined as $J = (j_1, j_2, \ldots, j_n)$. In this terminology, we define $f_L^J(s)$ to be equal

$$f_L^J(s) = \sum_{k=1}^{n} s_{j_k} \left(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})\right)$$

so it is easy to see that $f_L(s) = f_L^J(s)$ for J being the ordering $J^* = (j_1^*, j_2^*, \ldots, j_n^*)$ where $s_{j_1^*} \geq s_{j_2^*} \geq \cdots \geq s_{j_n^*}$. From Proposition 3.2 in [1], we know that if f is submodular then, $f_L^J(s)$ attains a maximum when $J = J^*$, i.e.,

$$f_L(s) \geq f_L^J(s) \qquad \forall \text{ ordering } J$$

Applying this observation to Equation(1), we get that

$$\begin{aligned} f_L(s) &= \lambda \sum_{k=1}^{n} s_{1,j_k}(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})) + (1-\lambda) \sum_{k=1}^{n} s_{2,j_k}(f(\{j_1, \cdots, j_k\}) - f(\{j_1, \cdots, j_{k-1}\})) \\ &\leq \lambda f_L(s_1) + (1-\lambda) f_L(s_2) \end{aligned}$$

which concludes the first direction of the proof.

Now for the second direction, assume that f_L is convex, and consider any two sets $A, B \subseteq V$ and let $\mathbb{1}_A, \mathbb{1}_B$ be the respective indicator vectors. In order to prove that f is submodular in this case, we will prove that $f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$. To see why this holds, consider the two sets $A \cup B$ and $A \cap B$. An easy observation yields that for any $i \in [n]$

$$(\mathbb{1}_{A \cup B} + \mathbb{1}_{A \cap B})_i = (\mathbb{1}_A + \mathbb{1}_B)_i = \begin{cases} 2 & \text{if } i \in A \cap B \\ 1 & \text{if } i \in A \triangle B \\ 0 & \text{if } i \notin A \cup B \end{cases}$$

Hence from Definition 12 we get that

$$\begin{aligned} f_L(\mathbb{1}_{A \cup B} + \mathbb{1}_{A \cap B}) &= 2(f(A \cap B) - f(\emptyset)) + 1(f(A \Delta B \cup A \cap B) - f(A \cap B)) \\ &= f(A \cap B) + f(A \cup B) \\ f_L(\mathbb{1}_A) &= f(A) \\ f_L(\mathbb{1}_B) &= f(B) \end{aligned}$$

But this means that

$$f_L(\mathbb{1}_{A \cup B} + \mathbb{1}_{A \cap B}) = f_L(\mathbb{1}_A + \mathbb{1}_B) \leq f_L(\mathbb{1}_A) + f_L(\mathbb{1}_B)$$

since f_L is convex by our assumption. Combining this all together we get that

$$f(A \cap B) + f(A \cup B) = f_L(\mathbb{1}_{A \cup B} + \mathbb{1}_{A \cap B}) \leq f_L(\mathbb{1}_A) + f_L(\mathbb{1}_B) = f(A) + f(B)$$

which yields the proof. □

Before we proceed, we give some examples of submodular functions, along with their Lovász extensions.

Example 1. $f(S) = |S|$.

It is easy to see that f is modular, since for any $S \subseteq T \subseteq V$ and $e \notin T$, we have that $\Delta(S|e) = \Delta(T|e) = 1$. Plugging in f into Definition 12, we get that for all $s \in [0,1]^n$, $f_L(s) = \mathbb{1}^T s$.

Example 2. For a $G \subseteq V$, $f(S) = \mathbb{1}_{G \cap S \neq \emptyset}(S)$, i.e, $f(S) = 1$ if $G \cap S \neq \emptyset$.

First, it is easy to see that f is monotone, hence for any $S \subseteq T \subseteq V$, we have that $f(T) \geq f(S)$. Now consider any $e \notin T$, then we have that $f(S \cup \{e\}) = f(T \cup \{e\}) = 1$. Now it follows directly from the monotonicity of f that $\Delta(S|e) \geq \Delta(T|e)$.Plugging in f into Definition 12, we get that for all $s \in [0,1]^n$,

$$f_L(s) = s_{j_1} \mathbb{1}_{j_1 \in G} + s_{j_2} \mathbb{1}_{j_2 \in G, j_1 \notin G} + \cdots + s_{j_n} \mathbb{1}_{j_n \in G, V \setminus j_n \notin G} = \max_{k \in G} s_k,$$

where $s_{j_1} \geq s_{j_2} \geq \cdots \geq s_{j_n}$.

Example 3. $f(S) = \sum_{G \in \mathcal{G}} d_G \mathbb{1}_{G \cap S \neq \emptyset}(S)$, where $d_G > 0$.

Since the function f of this example if a non-negative combination of functions from Example 2, it follows directly that f is submodular and

$$f_L(s) = \sum_{G \in \mathcal{G}} d_G \max_{k \in G} s_k.$$

3.2 Minimizing submodular functions

As mentioned earlier, the motivation behind defining the Lovász extension f_L for submodular functions is that we want to be able to minimize submodular function efficiently. We argued in the previous section that this indeed helpful as the extension we get is actually convex, and hence we know how to solve such problems efficiently. Moreover, we will show in this section that the extension is *very close* to the original function, in the sense that both f and f_L have the same minimum over their respective domains. More formally, we prove the following theorem.

Theorem 3.2 (SFMin is equivalent to a convex problem)**.** *Given a* normalized *submodular function f and its Lovász extension f_L,*

$$\min_{s \in \{0,1\}^n} f(s) = \min_{x \in [0,1]^n} f_L(x)$$

where any minimizer $s^\star$ of the LHS is a minimizer of the RHS, and any minimizer $x^\star$ of the RHS, has all its thresholded sets $\{x^\star \geq \theta\}, \forall \theta \in (0,1)$ as minimizers of the LHS.

Proof. We only prove the first part of the claim here, and the proof of the second part can be found in [1]. Namely, we only prove that

$$\min_{s \in \{0,1\}^n} f(s) = \min_{x \in [0,1]^n} f_L(x)$$

Let $s^\star$ and $x^\star$ be defined as in the statement of the theorem, then it is easy to see that $f(s^\star) \geq f(x^\star)$ since $\{0,1\}^n \subset [0,1]^n$. To prove the other direction (i.e., $f(s^\star) \leq f(x^\star)$), we resort to an equivalent definition of the Lovász extension as it appears in [1] and get the following:

$$\begin{aligned}
f_L(x^\star) &= \sum_{k=1}^{n-1} (x^\star_{j_k} - x^\star_{j_{k+1}}) f(\{j_1, \cdots, j_k\}) + x^\star_{j_n} f(V) \\
&\geq \sum_{k=1}^{n-1} (x^\star_{j_k} - x^\star_{j_{k+1}}) f(s^\star) + x^\star_{j_n} f(s^\star) && (x_{j_k} - x_{j_{k+1}} \geq 0) \\
&\geq x_{j_1} f(s^\star) \\
&\geq f(s^\star) && (\text{since } f(s^\star) \leq f(\emptyset) = 0 \text{ and } x_{j_1} \leq 1)
\end{aligned}$$

where $x^\star_{j_1} \geq x^\star_{j_2} \geq \cdots \geq x^\star_{j_n}$. □

We have thus far reduced the seemingly complex problem of minimizing a submodular function, to the easier problem of minimizing a continuous convex function f_L, i,e, solving $\min_{x \in [0,1]^n} f_L(x)$. This problem has been studied efficiently in the literature, and have many efficient algorithms, such as

- Minimum Norm Point (MNP) algorithm [2]: The running time of this algorithm is $O((n^3 EO + n^3 \log n + n^4) F^2)$, where $F := \max_i \{|f(\{i\})|, |f(V) - f(V \setminus i)|\}$, and EO is the running time of the evaluation oracle. The drawback of this algorithm is that it is weakly polynomial, as its running time depends on F, which is generally not desirable.

- Frank Wolfe algorithm [1]: The running time of this algorithm is $O((n^3 EO + n^3 \log n) F^2)$, hence it is also weakly polynomial. Although asymptotically its running time is better than that of MNP (n^3 versus n^4), the latter works faster in practice.

- Combinatorial algorithms, e.g., [8]: The running time of this algorithm is $O(n^5 EO + n^6)$ without any dependency on F, and hence is strongly polynomial.
- Cutting plane method [5]: If one is willing to settle for a weakly polynomial time algorithm, then the cutting plane method can be made to run in $O(n^2 \log nF \cdot EO + n^3 log^{O(1)} nF)$, otherwise its running time is $O(n^3 log^2 n \cdot EO + n^4 log^{O(1)} n)$.

4 Integer Linear Program

In this section, we study Integer Programs (IP) that basically be used to solve many interesting combinatorial problems. However since solving such problems is NP-hard, we study their relaxations as Linear Programs (LP) that can be solved efficiently, and discuss cases where the solution of the LP is the same as that of the IP. Integer Linear Programs (ILP) play a crucial role in Computer Science, as many interesting discrete optimization problems can be formulated as ILPs in the following way:

$$\beta^\natural \in \arg\max_{\beta \in \mathbb{Z}^m} \{\theta^T \beta : M\beta \leq c\} \tag{IP}$$

However, solving ILPs exactly is an NP-hard problem and that is listed in Karp's 21 NP-complete problems list [4]. Hence, the usual way to get around this issue is to relax the integrality constraints, and settle for a fractional solution with the hope of being able to round this fractional solution later on to an integral one, without *loosing too much*.

$$\beta^\star \in \arg\max_{\beta \in \mathbb{R}^m} \{\theta^T \beta : M\beta \leq c\} \tag{LP}$$

It is not hard to see that the optimal value of a relaxation is an upper bound on the optimal of the original ILP, as the domain of the latter is a subset of the domain of the former. We illustrate this in Figure 1, where the left figure shows the optimal value of an integral solution (i.e., that of ILP), and the right one shows the optimal solution of the relaxation (i.e, that of the LP).

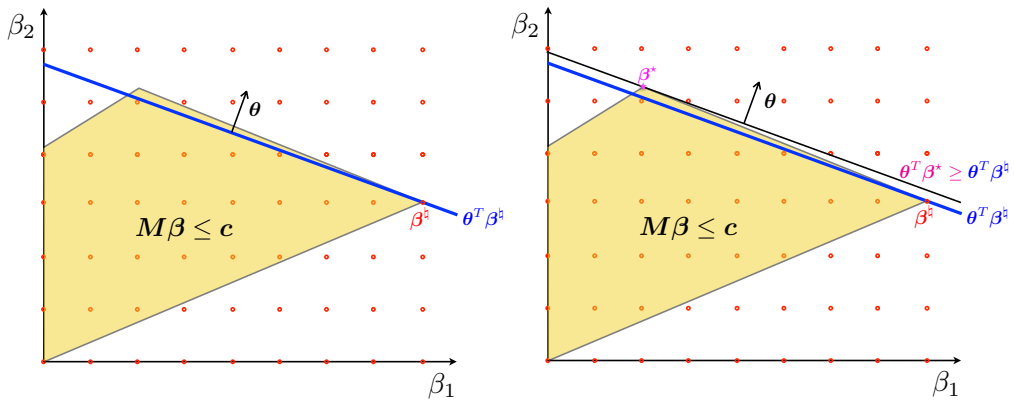

Figure 1: Optimal solution of ILP on the left. Optimal solution of LP on the right.

An important notion in the context of optimization is that of convex polyhedra, which is defined as $\mathcal{P} = \{\beta | M\beta \leq b\}$ for $\beta \in \mathbb{R}^m, c \in \mathbb{R}^m$. One can show that an optimal value for any linear objective function defined over $\mathcal{P}$ is attained at a vertex of $\mathcal{P}$, hence if we could prove that all vertices of our polyhedra of interest are integral, we would be able to solve our integer program to optimality, as any optimal value is in fact integral. We illustrate that in Figure 4.

We do know how to optimize linear function over such convex bodies efficiently up to arbitrary, where for instance we can use the interior point method that performs $O\left(\sqrt{l} \log \frac{l}{\epsilon}\right)$ iterations ($l > m$) with up to $O(m^2 l)$ operations, where ϵ is the absolute solution accuracy, and l is the number of constraints.

As mentioned earlier, if we could prove that our convex polyhedra of interest has only integral vertices, then we can solve our problem easily. In order to get this property, we study the structure of matrix M represents our constraints and defines our polyhedra. As it turns out, its structure would be crucial in order to solve its underlying optimization problems. A property that will be of special interest for us is that of Total Unimodularity defined as follows:

Definition 13 (Total unimodularity)**.** *A matrix $M \in \mathbb{R}^{l \times m}$ is totally unimodular (TU) iff the determinant of every square submatrix of M is* 0*, or* ± 1.

It is easy to see from the definition that all the entries of M are necessarily $0, \pm 1$, as the 1×1 sub matrices should also satisfy the properties of Definition 13. Moreover, given a matrix M, we can verify whether M is TU or not in polynomial time [10]. Before we proceed, we state some obversations about TU matrices. Namely, we give some operations that when applied to a TU matrix M, the resulting matrix remains TU.

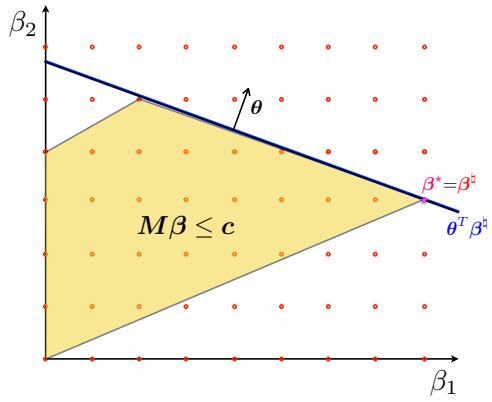

Figure 2: the vertices of $\mathcal{P}$ are integral.

Observation 1. *If M is TU then,*

1. *The tranpose of M is TU.*
2. *$M' = (M, I)$ is TU, where I is the identity matrix.*
3. *Matrix M' resulting from deleting a row (or a column) with at most one non-zero entry from M is TU.*
4. *Matrix M' resulting from interchanging two rows (or columns) in M is TU.*
5. *Matrix M' resulting from multiplying a row (or a column) from M by (-1) is TU.*
6. *Matrix M' resulting from duplicating rows (or columns) of M is TU.*
7. *Matrix M' resulting from applying a pivot operation[1] on M is TU.*

The reason why we are interested in our constraint matrix M being TU is that it leads to interesting properties for the underlying polyhedron. Formally,

Theorem 4.1 (TU Polyhedron is integral)**.** *The polyhedron $\mathcal{P} = \{M\beta \leq c\}$ has integer vertices when M is TU and c is an integer vector.*

Proof. Let z be any vertex of our polyhedron $\mathcal{P}$, then z is determined by a subsystem $M'z = c'$ where M' is a matrix of full row rank $m' \leq n$. We can now rewrite $M' = [U, V]$ where U is a $m' \times m'$ matrix of full row and column rank, and hence has a determinant ± 1 as it is sub matrix of M and it is non-singular. This says that z can be written as

$$z = \begin{bmatrix} U^{-1}c' \\ 0 \end{bmatrix}$$

From Cramer's rule, we know that $U^{-1} = \frac{\text{Adj}U}{\det U}$ where Adj$U$ is the adjugate matrix of U. But since $\det U \in \{\pm 1\}$ and Adj$U$ is an integer matrix and c' is an integer vector, we get that z is an integer vector as well. □

An equivalent definition of a TU matrix is the following:

Definition 14 (Equivalent definition of TU [9, 7])**.** *A matrix M is TU iff for every subset J of its columns, there exits a partition J_1, J_2 of J such that*

$$|\sum_{j \in J_1} M_{ij} - \sum_{j \in J_2} M_{ij}| \leq 1, \forall i = 1, \cdots, l$$

We give below examples of useful matrices in graph theory, that turn out to be TU.

Example 4. The edge-node incidence matrix T of a directed graph $G = (V, E)$.

[1] A pivot operation is the operation done in the reduced Gaussian elimination. For a $(0, \pm 1)$ matrix A, given a pivot $|a_{ij}| = 1$, if $a_{ij} = -1$, then we multiply the ith row by -1, and denote the new row by $\bar{a}^i$. For each row $k \neq i$, we modify it as follows:

$$\bar{a}^k = \begin{cases} a^k & \text{if } a_{kj} = 0 \\ a^k - \bar{a}^i & \text{if } a_{kj} = 1 \\ a^k + \bar{a}^i & \text{if } a_{kj} = -1 \end{cases}$$

Recall that if T is an $m \times n$ matrix where $m = |E|$ and $n = |V|$, and for each edge $e = (i, j) \in E$, $T_{e,i} = 1$ and $T_{e,j} = -1$. Any row M_k for $k \in \{1, 2, \ldots, m\}$, corresponds to an edge $e = (i, j)$ and hence has two non-zero entries (more precisely, a (1)-entry and a (-1)-entry) corresponding to the end points on this edge. Thus, given any subset J of the columns of T, let

$$J_1 = \{j \in J : \exists i \in [m] \text{ such that } T_{ij} \neq 0\}$$
$$J_2 = J \backslash J_1$$

Given that each row contains at 2 non-zero entries that are 1 and -1, we get that $|\sum_{j \in J_1} T_{i,j}| \leq 1$ for all $i \in [m]$. We also get by construction that $|\sum_{j \in J_2} T_{i,j}| = 0$, which proves that T is TU.

Example 5. The edge-node incidence matrix T of an undirected bipartite graph $G = (U, V, E)$ with bipartitions U and V.

When G is an unweighted graph, we define T such that $T_{e,i} = T_{e,j} = 1$ for any edge $e = (i, j) \in E$. Now given any subset J of columns corresponding to vertices in $U \cup V$, we define J_1, J_2 as follows:

$$J_1 = J \cap U \qquad \text{and} \qquad J_2 = J \cap V$$

Since G is bipartite, it is easy to see that no edge can have both his endpoints neither in J_1 nor in J_2, hence for all $e \in [m]$, we have

$$\sum_{j \in J_1} T_{e,j} \in \{0, 1\} \qquad \text{and} \qquad \sum_{j \in J_2} T_{e,j} \in \{0, 1\}$$

which proves that T is TU.

Example 6. The biadjacency matrix B of an undirected acyclic bipartite graph $G = (U, V, E)$, such that the degree of every node in V is at most 2.

Recall that the biadjacency matrix B is an $r \times s$ matrix where $r = |V|$, $s = |U|$ such that $B_{v,u} = 1$ if and only if $v \in V, u \in U$ and $(u, v) \in E$. To see why B is TU, consider another graph $G' = (U, E')$ over the vertices in the right partition U, such that $e = (u_1, u_2) \in E'$ iff there exists a vertex $v \in V$ such that $(u_1, v), (u_2, v) \in E$, i.e., if u_1 and u_2 share a common neighbour in the original graph. It is not hard to see that G' is acyclic and hence bipartite over some partitions U_1, U_2; G' cannot have a cycle of length 3 as this would imply that exists a vertex in V whose degree is greater than 2. Similarly, a cycle of length $k \geq 4$ would correspond to a cycle of length $2k$ in G, but G is acyclic.

Now in order to prove that B is TU, we provide a way to partition any subset J of the columns of J such that the TU condition is satisfied. Recall that the columns if B are indexed by vertices in U, the rows are indexed by vertices in V, and hence each row has at most two non-zero entries. Given a subset J of the columns of B, we define J_1, J_2 as follows:

$$J_1 = J \cap U_1 \qquad \text{and} \qquad J_2 = J \cap U_2$$

where U_1, U_2 are the bipartition's of the new graph G' defined earlier. Now for every row indexed $v \in V$, we have two cases:

1. v has only one neighbour in U, hence the row has only one nonzero entry, so the TU condition is satisfied trivially.
2. v has two neighbours $u_1, u_2 \in U$. Note that in this case we have an edge $(u_1, u_2) \in E'$, and since G' is bipartite we get that $|U_1 \cap \{u_1, u_2\}| = |U_2 \cap \{u_1, u_2\}| = 1$, and hence

$$\sum_{u \in J_1} B_{v,u} \in \{0, 1\} \qquad \text{and} \qquad \sum_{u \in J_2} B_{v,u} \in \{0, 1\}$$

and hence the TU condition is satisfied.

We present below another application where TU matrices turn out to be handy. In this problem, we are given a vector $y \in \mathbb{R}^n$ and an integer k, and the goal is to find a k sparse vector $x \in \mathbb{R}^n$ that best approximates y in the ℓ_2 sense. More formally the problem of interest, known as the k-sparse projection, can be stated as follows:

Problem 1 (k-sparse projection)**.** *The projection of a vector $y \in \mathbb{R}^n$ over the set of k-sparse vectors is given by:*

$$\min_{x \in \mathbb{R}^n} \{||x - y||_2^2 : ||x||_0 \leq k\}$$

where $||x||_0 = |\text{supp}(x)|$ *where* $\text{supp}(x) = \{i : x_i \neq 0\}$.

We will see now how to cast this problem as an maximization problem where the underlying matrix is TU. To see that note that our problem can be equivalently rewritten as:

$$\min_{x \in \mathbb{R}^n} \{||x - y||_2^2 : ||x||_0 \leq k\} = \min_{|S| \leq k} \min_{x \in \mathbb{R}^n} \{||x - y||_2^2 : \text{supp}(x) = S\}$$

For a fixed $|S| \leq k$, and an $x \in \mathbb{R}^n$ such that $\text{supp}(x) = S$, we can break the summation of the ℓ_2 norm as

$$\begin{aligned} \|x - y\|_2^2 = \sum_{i \in S} (x_i - y_i)^2 + \sum_{i \notin S} y_i^2 \\ = \|y\|_2^2 - \max_{|S| \leq k} \{ \sum_{i \in S} y_i^2 \} \end{aligned}$$

where the first term can be brought down to zero by setting $x_i = y_i$ for all $i \in S$. Hence the problem boils down to finding the highest energy k terms in k, i.e.,

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \{ \|x - y\|_2^2 : \|x\|_0 \leq k \} = \min_{|S| \leq k} \{ \sum_{i \notin S} y_i^2 \} \\ = \|y\|_2^2 - \max_{|S| \leq k} \{ \sum_{i \in S} y_i^2 \} \end{aligned}$$

But we can view $\max_{|S| \leq k} \sum_{i \in S} y_i^2$ as an integer program over TU constraints:

$$\max_{s \in [0,1]} \{ \sum_{i=1}^{n} s_i y_i^2 : \mathbb{1}^T s \leq k \} = \max_{s \in [0,1]} \{ \sum_{i=1}^{n} s_i y_i^2 : \mathbb{1}^T s \leq k \}$$

Since matrix $\mathbb{1}^T$ is TU and $k \in \mathbb{Z}$.

References

[1] Francis Bach. Learning with submodular functions: A convex optimization perspective. *arXiv preprint arXiv:1111.6453*, 2011.

[2] Deeparnab Chakrabarty, Prateek Jain, and Pravesh Kothari. Provable submodular minimization via fujishige-wolfe's algorithm.

[3] Uriel Feige, Vahab S Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[4] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[5] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1049–1065. IEEE, 2015.

[6] László Lovász. Submodular functions and convexity. In *Mathematical Programming The State of the Art*, pages 235–257. Springer, 1983.

[7] George L Nemhauser and Laurence A Wolsey. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988.

[8] James B Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009.

[9] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[10] Klaus Truemper. Alpha-balanced graphs and matrices and gf (3)-representability of matroids. *Journal of Combinatorial Theory, Series B*, 32(2):112–139, 1982.

[11] Jan Vondrák. Continuous extensions of submodular functionscontinuous extensions of submodular functions. CS 369P: Polyhedral techniques in combinatorial optimization, http://theory.stanford.edu/ jvondrak/CS369P-files/lec17.pdf, 2010.