AN INTRODUCTION TO SUBMODULAR FUNCTIONS

ANDREA ARALDO

Abstract. In this tutorial, we will review in a simple way the use of submodular functions as a tool to solve optimization problems. A greedy algorithm is given and its $\frac{1}{1-1/e}$ approximation to the optimum is proved. Recognizing if an optimization problem can be described in this framework can be very useful as it permits to directly use the many results that have been already found in the field (one of them is the approximation bound of greedy algorithms).

1. Outline

- We provide the required definitions and some examples that fit into them.
- We describe a greedy algorithm and give a proof of its $\frac{1}{1-1/e}$ approximation (It means the the utility of the solution provided by the greedy algorithm is at least 63% of the optimal one).
- We provide an example to show how to recognize if a problem can be described by means of submodular functions.

2. Definitions

Let us consider a set A, that we will call *ground set*, and a function $f : 2^A \to \mathbb{R}$, i.e. a function that associate to a subset $S \subseteq A$ a real value $f(S)$. Sometimes we will call a subset of A a collection.

Definition 1. f is *monotone* non decreasing if

$$\forall S \subseteq T \subseteq A \Rightarrow f(S) \leq f(T)$$
$$f(\emptyset) = 0$$

We will consider only this kind of functions, even when not explicitly specified.

Definition 2. Given a function f, we associate to it another function $\rho(i|S)$, $\forall i \in A, S \subseteq A$, that we call the gain of adding the element i to the collection S:

$$\rho(i|S) \triangleq f(S \cup \{i\}) - f(S)$$

Definition 3. f is *modular* if its associated gain is such that

$$\rho(i|S) = f(\{i\}),\ \forall S \subseteq A, \forall i \in A \setminus S$$

f is *submodular* if

$$\rho(i|S) \geq \rho(i|T),\ \forall S \subseteq T \subseteq A, \forall i \in A \setminus S$$

f is supermodular if

$$\rho(i|S) \leq \rho(i|T),\ \forall S \subseteq T \subseteq A, \forall i \in A \setminus S$$

Date: 08/10/2015.

Key words and phrases. optimization; submodular functions; algorithms; approximation algorithms.

Proposition 4. *The previous definitions are equivalent to*

$$\begin{cases} f(S) = \sum_{i \in S} f(\{i\}) & \textit{iff } f \textit{ is modular} \\ f(S) \leq \sum_{i \in S} f(\{i\}) & \textit{iff } f \textit{ is submodular} \\ f(S) \geq \sum_{i \in S} f(\{i\}) & \textit{iff } f \textit{ is supermodular} \end{cases}$$

To resume the meaning of the previous definitions, we can say that, if f is modular, the value of a collection is the sum of the individual values, while if f is submodular the value of a collection can be also less and if f is supermodular it can also be more.

Example 5. We now give some high level example of each of the functions above.

MODULAR FUNCTION. Let us consider a set of people A. The value of each person i is $f(\{i\})$, the amount of money he has in his pocket. For each collection of people $S \subseteq A$ we compute the value $f(S)$ as the amount of money that the people in the collection collectively have. f is modular.

SUBMODULAR FUNCTION. You are an organizer of a "language exchange café" and you value a person i with $f(\{i\})$, i.e. the number of languages he can speak. The value $f(S)$ of a collection $S \subseteq A$ is the number of languages that are collectively spoken. f is submodular, because when adding a new person i to the collection S, some of the languages he can speak may be already spoken by other people in S. Therefore the gain can be less than the number of languages of the new person. If you consider two collections $S \subseteq T$, than the contribution given by the new person to the small set makes more difference with respect to when the person is added to a superset T, since it is more likely that most of the languages spoken by the person are already covered in T.

SUPERMODULAR FUNCTION. You provide a social network service and you put inside each message set by a user an advertisement. Therefore, you value a user $i \in A$ as the number $f(\{i\})$ of messages she sends. The value $f(S)$ of a collection of your users S is the number of messages that are collectively sent. Of course, adding a user to a big set of pre-existing users brings a bigger gain with respect to adding her to a smaller set, as in the big collection there will be more users willing to communicate with her and to whom she can be interested in talking.

Remark 6. Submodularity embeds the concept of diminishing return, used a lot in economy. Supermodularity can be useful when studying interactions among agents.

In this tutorial, we will only focus on submodular functions.

Definition 7. A set of collections $\mathcal{M}$ is a *uniform matroid* if there exists a number $k \in \mathbb{N}$ such that

$$\mathcal{M} = \{S \subseteq A | \, |S| \leq k\}$$

Sometimes, we will call the collections in $\mathcal{M}$ the *feasible collections*.

Example 8. Coming back to the "language café" example, suppose you are organizing your event in a bar that can host k people at most. Therefore you cannot choose whatever collection, because the feasible ones are only those whose cardinality is less or equal to k.

In this tutorial we are interested in solving the following optimization problem.

Problem 9. Given a submodular function f and a matroid $\mathcal{M}$, we want to maximize f finding the best feasible collection.

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

subject to

$$S \in \mathcal{M}$$

3. Algorithm

How to solve the problem above? In theory we should consider all the possible subsets of A (they are $2^{|A|}$) and, for each of them, see if it is feasible and compute the value of f. Of course this is too expensive.

Theorem 10. *The problem above is NP complete*

We will prove this result later. We now describe a simple greedy algorithm which gives a solution of the problem. We will prove later that its gap w.r.t. to the optimal solution is bounded.

Algorithm 1: Greedy algorithm

Input : $A, f : 2^A \to \mathbb{R}, \mathcal{M}$
Output: $S_{alg} \in \mathcal{M}$
$S_0 = \emptyset$
for $j = 1$; $j \leq K$; $j{+}{+}$ **do**
  $i^* = \arg\max_{i \in A \setminus S_{j-1}} \rho(i|S_{j-1})$
  $S_j = S_{j-1} \cup \{i^*\}$
end
$S_{alg} = S_k$

The greedy algorithm constructs the collection adding at each iteration, the object that brings the most value to the collection constructed so far. This algorithm has a polynomial time complexity, since the arg max calculation can be implemented by means of ordering that has a polynomial time cost.

Our goal is now to show that the value $f(S_{alg})$ has a bounded distance w.r.t. to the optimal value $f(S_{opt})$. We first need the following lemma.

Lemma 11. *If f is a non decreasing submodular function and A is the ground set, then for every $S, T \subseteq A$*

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} \rho(j|S)$$

Proof. Since f is non-decreasing, $f(T) \leq f(T \cup S)$. For the sake of simplicity, let us enumerate the elements in $T \setminus S = \{a_1, \ldots, a_n\}$ and observe that $T \cup S = S \cup \bigcup_{i=1}^{n} \{a_n\}$. Therefore

$$f(T) \leq f\left(S \cup \bigcup_{i=1}^{n} \{a_i\}\right)$$

We can build $S \cup \bigcup_{i=1}^{n}\{a_i\}$ incrementally, by adding 1 to S and obtaining a marginal utility $p(a_1|S)$, than by adding a_2 to $S \cup \{a_1\}$ and obtaining the marginal utility $\rho\left(a_2|S \cup \{a_1\}\right)$ and so on. Therefore

$$f\left(S \cup \bigcup_{i=1}^{n}\{a_i\}\right) = f(S)+\rho(a_1|S)+\rho\left(a_2|S \cup \{a_1\}\right)+\cdots+\rho\left(n|S \cup \{a_1\}\ldots\{a_{n-1}\}\right)$$

Observe that, for the submodularity, $\rho\left(i|S \cup \{a_1\}\ldots\{a_{i-1}\}\right) \leq \rho\left(a_i|S\right)$. Therefore

$$f\left(S \cup \bigcup_{i=1}^{n}\{a_i\}\right) \leq f(S) + \sum_{i=1}^{n}\rho(a_i|S)$$

This proves the lemma. □

We can finally prove our theorem.

Theorem 12. *The greedy algorithm is a 1/(1-1/e) approximation w.r.t. Optimum*

Proof. Let us denote with T the optimal set of elements, of A, i.e. the one that maximizes the utility subject to the matroid constraint, and be $z_{opt} = f(T)$ the optimal utility. Let us denote with z_{alg} the utility of the greedy algorithm. We have to show that $z_{alg} \geq (1-1/e)z_{opt}$. It is known that $\left(1-\frac{1}{k}\right)^k \leq 1/e$, $\forall k \in \mathbb{N}$. Therefore $1-\left(1-\frac{1}{k}\right)^k \geq 1-1/e$. Therefore, in order to show the thesis, it suffices to show that

$$z_g \geq \frac{k^k-(k-1)^k}{k^k}z_{opt} \tag{3.1}$$

for at least one value $k \in \mathbb{N}$. We choose k equal to the maximum cardinality of a feasible set, that corresponds also to the number of iterations of the algorithm. After the j-th insertion, the set built by the greedy algorithm is S_j. If we show that

$$f(S_j) \geq \frac{k^j-(k-1)^j}{k^j}z_{opt}, j = 1\ldots,k \tag{3.2}$$

we are sure that Equation 3.1 holds and the thesis will be proved.

We want to prove the inequality above by induction. Observe that it holds for $j = 0$. Now, suppose it holds for a certain j. We want to prove that it holds for $j+1$.

Observe that, thanks to the lemma above, at each step j:

$$f(T) \leq f(S_j) + \sum_{i \in T \setminus S_j}\rho(i|S_j) \tag{3.3}$$

We will find upper bounds to the two terms of the sum above. Let us denote ρ_i the marginal utility achieved by the greedy algorithm at step i, i.e. $\rho_i \triangleq f(S_i) - f(S_{i-1})$. Since the greedy algorithm selects at each step the element that brings the maximum marginal utility, we are sure that $\rho_{j+1} = \max_{i \in A \setminus S_j}\rho(i|S_j)$. Therefore $\rho_{j+1} \geq \rho(i|S_j), \forall i \in T \setminus S_j, j = 1\ldots k-1$ and $\sum_{i \in T \setminus S_j}\rho(i|S_j) \leq |T \setminus S_j| \cdot \rho_{j+1} \leq k\rho_{j+1}$. Thanks to what we found, we can rearrange (Equation 3.3) as

$$f(T) \leq f(S_j) + k\rho_{j+1}$$

$$\Rightarrow \rho_{j+1} \geq \frac{1}{k}z_{opt} - \frac{1}{k}f(S_j)$$

Adding $f(S_j)$ to both side

$$f(S_{j+1}) = f(S_j) + \rho_{j+1} + \geq \frac{1}{k} z_{opt} + \left(1 - \frac{1}{k}\right) f(S_j), j = 1 \dots k-1$$

We now substitute the induction hypothesis Equation 3.2 inside the equation above

$$\sum_{i=1}^{j+1} \rho_i \geq \frac{1}{k} z_{opt} + \left(1 - \frac{1}{k}\right) \frac{k^j - (k-1)^j}{k^j} z_{opt}, j = 1 \dots k-1$$

that is equivalent to

$$f(S_{j+1}) \geq \frac{k^{j+1} - (k-1)^{j+1}}{k^{j+1}} z_{opt}$$

□

4. Recognizing a submodular problem

When facing an optimization problem it may not be trivial to have some insight of its properties and to build approximation algorithms. But f one recognizes that the problem is submodular, one can simply exploit all the results available in submodular function theory without trying to build them from the scratch.

In this section we consider, as an example, the classic Maximum Coverage Problem.

Problem 13. We have a set $U = \{u_1, \dots u_N\}$, that we call universe, and a finite set of subsets of the universe $A = \{a_1, a_2, \dots, a_n\}$, where $a_i \subseteq U$. We want to find a collection $S \subseteq A$ of this subsets that covers as many elements of U as possible. We cannot consider as many subsets we want, but we are constrained to take at most k of them. The problem is usually formalized as an ILP problem.

We introduce a decision variable x_i for each subset a_i, where $x_i = 1$ if we include subset a_i in our collection, $x_i = 1$ otherwise. We also introduce variable y_j for each element u_j of the universe, which is one if u_j is covered, i.e. at least one of the selected subsets includes u_j. The problem is

$$\max \sum_{u_j \in U} y_j$$

subject to

$$\sum_{a_i \in A} x_i \leq k$$

$$\sum_{a_i \ni u_j} x_i \geq y_j, \forall u_j \in U$$

$$y_j \in \{0, 1\}, \forall u_j \in U$$

$$x_i \in \{0, 1\}, \forall a_i \in A$$

This problem is known to be NP hard. How can we build an approximate algorithm? How can we find its theoretical gap? If we blindly look at the formulation above these questions may be hard to answer. But if we can describe the problem above in terms of a submodular function and a matroid, we can use the general result given in the previous sections.

We consider A, i.e. the set of subsets of the universe, our ground set. We associate to a collection $S \subseteq A$ a utility that is the number of elements of the universe covered by the subsets considered in the collection, $f(S) = |\bigcup_{a \in A} a|$.

It is immediate to show that f is non decreasing. We want to show that it is submodular. Let us consider a subset of the universe $a \in A$. The gain of adding it to the collection S is the number of elements included in a that are not included in any of the subsets belonging to S:

$$\rho(a|S) = \left| a \setminus \left\{ \bigcup_{b \in S} b \right\} \right|$$

Taking two collection S, T such that $S \subseteq T$, some of the new elements of the universe that a is bringing to S may already be covered by some subset of the collection T. Therefore

$$\rho(a|S) \geq \rho(a|T)$$

This proves the submodularity of f. The fact that we can only consider collections of at most k subsets of the universe defines a matroid

$$\mathcal{M} = \{S \subseteq A| \, |S| \leq k\}$$

Therefore the problem can be expressed in this more compact form

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

subject to

$$S \in \mathcal{M}$$

We can then directly apply the greedy algorithm and we are guaranteed that its solution is at most 67% far from the optimum.

We profit from what said about the Maximum Coverage Problem to prove the following

Theorem 14. *The maximization of a submodular function f subject to a uniform matroid constraint is a NP hard problem.*

Proof. The problem we are talking about is whatever problem on the form of Problem 9. We have shown that the Maximum Coverage Problem is a particular instance of that problem. This proves the thesis. □

5. Extensions

Submodular functions can describe a vast set of problems, more general that the ones we have examined here. And in some specific cases, they can provide stronger results, like the following.

Theorem 15. *If function f is modular, then the greedy algorithm gives the exact optimal solution.*

We have just talked about uniform matroids, but similar results hold also in cases where constraints are more complex, for example when each element has a weight $w_i, \forall i \in A$ and the constraint is that the sum of the weights of the elements in the collection must not exceed a threshold, like in a knapsack problem. In this case, we have another type of matroid

$$\mathcal{M} = \{S \subseteq A| \sum_{i \in S} w_i \leq K\}$$

Results are similar but not identical. In case of non uniform matroid a simple greedy algorithm provides only a 2-approximation and more complex algorithms

exist to find a $\frac{1}{1-1/e}$ approximation. In general, when trying to formalize a problem, even if it cannot be described in the terms defined here, one should check the literature about submodular functions to see if useful results have been found.

6. Useful Links

An entire course on this subject has been proposed at University of Washington Bilmes [2014].

References

Jeff A. Bilmes. Submodular Functions, Optimization, and Applications to Machine Learning, 2014. URL `http://j.ee.washington.edu/{~}bilmes/classes/ee596b{_}spring{_}2014/`.

E-mail address: `araldo@lri.fr`