Submission and Formatting Instructions for International Conference on Machine Learning (ICML 2017)

Anonymous Authors[1]

Abstract

We will offer a new approach for solving the problem of two stage constrained submodular maximization. Our solution offers better results as well faster runtime compared to previous algorithms devised for this task. Historically, submodular maximization has represented a topic of interest in Computer Science and Machine Learning, especially in data summarization. The single stage submodular problem is solvable by a fast greedy algorithm with proven theoretical guarantees. The two stage problem adds a layer of complexity to this task, and was previously tackled in (**?**). We improve on their approach, offering a $(1 - 1/e)/2$ approximation ration for general monotone submodular functions.

1. Introduction

Our paper is focused on solving data summarization problems - given a set of elements (images, news articles, movies and their ratings) and several categories these elements are part of, we are interested in a selecting a subset of the original set such that each category is well represented in said subset. For our problem, both the subset we are selecting and the number of elements in the subset that we can use for each category will come under cardinality constraints.

TODO: Paragraph on history of submodularity research..

We consider the setup in (**?**). Let us consider we have a set of n elements, each element being assigned to certaing categories. There are m such categories, and we can measure the relelvance of a set with respect to a category through m submodular functions, each category having a corresponding function. We first want to pick a ground set S of size at most l from the n elements at our disposal. Then, for each category, we will be able to choose a set $S_i \subset S$ such that S_i has size at most k. Hence the two stage problem.

The setup for this problem allows us to tackle a much wider gamut of problems compared to the single stage constrained submodular maximization. Some examples, which we will later describe in this paper, include image processing (finding a subset of images most relevant to certaing features), movie recommendation (based on user preference, what movies can we select such that we make a good selection of family, adventure, horror etc. films) or article summarization (given a corpus of articles, find a subset that are relevant to several news categories). When running these experiemnts, we will compare performance to the algorithm described in (**?**), and show that our approach is superior in runtime and similar, if not better in objective value.

TODO: Paragraph about structure of paper

2. Problem Setup

Let us formally define the objective function F we are trying to maximize. Consider our elements are part of the universe Ω. We are interested in

$$\max_{\substack{S \subset \Omega \\ |S| \leq l}} \sum_{i=1}^{m} \max_{\substack{T : T \subset S \\ |T| \leq k}} f_i(T)$$

We define our objective value as $F(S) = \sum_{i=1}^{m} \max_{\substack{T : T \subset S \\ |T| \leq k}} f_i(T)$. All functions f_i must be monotone normalized submodular functions.

TODO: too similar to sparse comb. rep??

TODO: talk about why objective is not submodular?

[*]Equal contribution [1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

3. Algorithm

Below we provide our solution to the two stage problem. Our algorithm works in an incremental manner, it starts out with an empty set S, and then fills it up with k elements in a greedy fashion. Then, we do $l - k$ more iterations to pick the rest of the elements, and we update the sets T_i along the process. Our approach should perform $O(lnmk)$ function evaluations.

TODO: rewrite this in the format from the email?

TODO2: proof (?), theorems, appendix

4. Experiment Setup

In the next section, we present empirical evidence for the performance of our algorithm. Here we describe the four other routines we test against alongside the datasets we considered.

4.1. Routines

4.1.1. Our algorithm

TODO: should we mention this?

4.1.2. Local Search

This is the algorithm used in (**?**).

TODO: sketch it here?? How detailed?

4.1.3. Greedy Sum

We will greedily pick the element that maximizes our objective function as though the cardinality constraint in the second stage were l. In other words, at each one of the l steps we'll maximize

$$\sum_{i=1}^{m} \max_{\substack{x \in \Omega \\ x \notin S}} f_i(S \cup \{x\}) - \sum_{i=1}^{m} f_i(S)$$

At the end of this procedure we have $T_1 = \cdots = T_m = S$. To reinstate the k cardinality constraint in the second stage, we will perform submodular maximization on the elements in S for each of the m categories.

4.1.4. Greedy Merge

We will ingore the first stage constraint on l, and perform constrained submodular maximization for each function subject to $T_i \leq k$. Our solution set S will be the union of all the T_i. The point of this approach is that even if we fail to obtain a set of cardinality at most l, we do however get a good estimate for an upper bound. This is not a tight upper bound however, as the greedy solution only guarantees a $1 - 1/e$ approximation of the optimal (TODO citation). Thus, for large values of l this approach can be outperformed by the other routines.

4.1.5. Clustering

TODO: we might want to omit due to poor perf?

We perform *k-means* clustering. We pick l cluster centers, in the idea that these would give a good summary of our data. We then perform single stage constrained submodular maximization and pick elements for each set T_i from these l cluster centers.

4.2. Datasets and choices for *f*

TODO: should probably make these more detailed. On the right track? need links to datasets and citations in bib.

4.2.1. Wikipedia

We run our code alongside the code from (**?**) on a Wikipedia ML dataset. For this experiment we choose f to be coverage functions.

4.2.2. Movielens

We analyze part of the Movielens 100k (TODO check) dataset. In our experiments we set n, m, l, k to ????. We set our functions f to the *Facility Location* function from (TODO cite paper in foler). We compute the similarity between two movies based on a precomputed similarity matrix that we input to our algorithm. TODO: more detalis about this?

4.2.3. Image Processing

We consider part of the VOC2012 dataset. The dataset contains images associated with various tags. For each image, we create a feature vector out of the tags associated with said image, and use Facility Location as our function of choice. Similarity is defined as the $2-$norm between the feature vectors. We ran experiments for $n = 100, m = 20, l = 8, k = 5$. TODO: can rerun. right ballpark?

4.2.4. Article Summarization

We consider the reuters21578 dataset. This is an article database, each article having some keywords associated with it. We again use Facility Location, and compute similarity through cosine similarity. Our experiments have $n = 126, m = 35, l = 10, k = 5$.

TODO: Should probably standardize experiments somewhat?? At least l and k, as m is forced for image proc and art summarization (we could change it but is tedious and might not be worth it)

5. Experiment Analysis

5.1. Choices for the functions

This is up in the air. We need plots here, and discuss what's going on. What kind of plots do we want? Ideas

- L fixed increasing K
- K fixed increasing L
- Matching N, L, K over experiemnts?
- Plots of objective value only? We can plot runtime, at least for wiki experiment. That way we compare directly with their implementation and can't be accused of having things inteantionaly too slow on our end.
- ?

Also, add anything else?