Submodular Meta-Learning

Arman Adibi∗, Aryan Mokhtari†, Hamed Hassani‡

Abstract

In this paper, we introduce a discrete variant of the meta-learning framework. Meta-
learning aims at exploiting prior experience and data to improve performance on future
tasks. By now, there exist numerous formulations for meta-learning in the continuous do-
main. Notably, the Model-Agnostic Meta-Learning (MAML) formulation views each task
as a continuous optimization problem and based on prior data learns a suitable initializa-
tion that can be adapted to new, unseen tasks after a few simple gradient updates. Moti-
vated by this terminology, we propose a novel meta-learning framework in the discrete
domain where each task is equivalent to maximizing a set function under a cardinality
constraint. Our approach aims at using prior data, i.e., previously visited tasks, to train
a proper initial solution set that can be quickly adapted to a new task at a relatively low
computational cost. This approach leads to (i) a personalized solution for each individual
task, and (ii) significantly reduced computational cost at test time compared to the case
where the solution is fully optimized once the new task is revealed. The training proce-
dure is performed by solving a challenging discrete optimization problem for which we
present deterministic and randomized algorithms. In the case where the tasks are mono-
tone and submodular, we show strong theoretical guarantees for our proposed methods
even though the training objective may not be submodular. We also demonstrate the ef-
effectiveness of our framework on two real-world problem instances where we observe that
our methods lead to a significant reduction in computational complexity in solving the
new tasks while incurring a small performance loss compared to when the tasks are fully
optimized.
1 Introduction

Many applications in artificial intelligence necessitate exploiting prior data and experience to enhance quality and efficiency on new tasks. This is often manifested through a set of tasks given in the training phase from which we can learn a model or representation that can be later used for new unseen tasks in the test phase. In this regard, meta-learning aims at exploiting the data from the available tasks to learn model parameters or representation that can be later used to perform well on new unseen tasks, in particular, when we have access to limited data and computational power at the test time [TP12, Sch92, BBC90, VD02]. By now, there are several formulations for meta-learning, but perhaps one of the most successful ones is the Model-Agnostic Meta-Learning (MAML) framework proposed in [FAL17]. In MAML, we aim to train the model parameters such that applying a few steps of gradient-based updates with a small number of samples from a new task would perform well on that task. MAML can also be viewed as a way to provide a proper initialization, from which performance on a new task can be optimized after a few gradient-based updates. Alas, this scheme only applies to settings in which the decision variable belongs to a continuous domain and can be adjusted using gradient-based methods at the test time.

Our goal is to extend the methodology of MAML to the discrete setting. We consider a setting in which our decision variable is a discrete set, and our goal is to come up with a good initial set that can be quickly adjusted to perform well over a wide range of new tasks. In particular, we focus on submodular maximization to represent the tasks which is an essential class of discrete optimization.

There are numerous applications where the submodular meta-learning framework can be applied to find a personalized solution for each task while significantly reducing the computation load. In general, most recommendation tasks can be cast as an instance of this setting [GKW+13, EAVSG09, YG11]. Consider the task of recommending a set of items, e.g., products, locations, ads, to a set of users. One approach for solving such a problem is to find the subset of items that have the highest score over all the previously-visited users and recommend that subset to a new user. Indeed, this approach leads to a reasonable performance at test time; however, it does not provide a user-specific solution for a new user. Another approach is to find the whole subset at the test time when the new user arrives. In contrast to the previous approach, this scheme leads to a user-specific solution, but at the cost of running a computationally expensive algorithm to select all the elements at the test time.

In our meta-learning framework, the process of selecting set items to be recommended to a new user is done in two parts: In the first part, a set of items is selected offline according to prior experience. These items are the most popular items to the previously-visited users (depending on the context). In the second part, which happens at the test time, a set of items that is personalized to the coming user is selected. These are items that are computed specifically according to the features of the coming user. In this manner, the computation for each coming user would be reduced to the selection of the second part, which typically constitutes a small portion of the final set of recommended items. The first part can be done offline with a lower frequency. For instance, in a real recommender system, the first part can be computed once every hour, and the second part can be computed specifically for each coming user (or for a class of similar users). While we have mentioned recommendation (or more generally facility location) as a specific example, it is easy to see that this framework can be easily used to reduce computation in other notable applications of submodular optimization.
Contributions. Our contributions are threefold:

- We propose a novel discrete meta-learning framework where each task is equivalent to maximizing a set function under some cardinality constraint. Our framework aims at using prior data, i.e., previously visited tasks, to train a proper initial solution set that can be quickly adapted to a new task at a low computational cost to obtain a task-specific solution.

- We present computationally efficient deterministic and randomized meta-greedy algorithms to solve the resulting meta-learning problem. When the tasks are monotone and submodular, we prove that the solution obtained by the deterministic algorithm is at least 0.53-optimal, and the solution of the randomized algorithm is $(1 - 1/e - o(1))$-optimal in expectation, where the $o(1)$ term vanishes by the size of the solution. These guarantees are obtained by introducing new techniques, despite that the meta-learning objective is not submodular.

- We study the performance of our proposed meta-learning framework and algorithms for movie recommendation and ride-sharing problems. Our experiments illustrate that the solution of our proposed meta-learning scheme, which chooses a large portion of the solution in the training phase and a small portion adaptively at test time, is very close to the solution obtained by choosing the entire solution at the test time when a new task is revealed.

1.1 Related work

Continuous Meta-Learning. Meta-learning has gained considerable attention recently mainly due to its success in few shot learning [VBL+16, RL17, SSZ17, WY19] as well as reinforcement learning [DSC+16, WKNT+16, SGY+20, FMO20b]. One of the most successful forms of meta-learning is the gradient-based Model Agnostic Meta-learning (MAML) approach [FAL17]. MAML aims at learning an initialization that can be adapted to a new task after performing one (or a few) gradient-based update(s); see, e.g., [FMO20a]. This problem can be written as

$$\min_{w \in W} \mathbb{E}_{a \sim p} [f_a(w - \nabla f_a(w))],$$

where $W \subseteq \mathbb{R}^d$ is the feasible set and $l$ is the probability distribution over tasks. The previous works on MAML including [NAS18, FXL18, GFL+18, YKD+18, AES19, RFKL19, FMO20a, CMS20] consider the case where $W$ is a continuous space. In fact none of these works can be applied to the case where the feasible parameter space is discrete. In this paper, we aim to close this gap and extend the terminology of MAML to discrete settings.

Submodular Maximization. Submodular functions have become key concepts in numerous applications such as data summarization [LB11, WLKB13, KB14, MKSK16], viral marketing [KKT03], sensor placement [KSG08], dictionary learning [DK11], and influence maximization [KKT03]. It is well-known that for maximizing a monotone and submodular function under the cardinality constraint, the greedy algorithm provides a $(1 - 1/e)$-optimal solution [KG, NW78, Wol82]. There has been significant effort to improve the scalability and efficiency of the greedy algorithm using lazy, stochastic, and distributed methods [MBK+15, KLHK17, BENW15, MZ15, KMMV15, HSK17, MHK20, HKMS19, BRS19]. However, our framework is fundamentally different and complementary to these approaches as it proposes a
new approach to use data at training time to improve performance at new tasks. Indeed, all the aforementioned techniques can be readily used to further speed-up our algorithms. Optimization of related submodular tasks has been a well-studied problem with works on structured prediction [LB12], submodular bandits [YG11, ZCHK19], online submodular optimization [JB11, SG09, GKS14, CHHK18], and public-private data summarization [MZK16]. However, unlike our work, these approaches are not concerned with train-test phases for optimization. Another recently-developed methodology to reduce computation is the two-stage submodular optimization framework [BMKS16, MKZK18, SZKK17], which aims at summarizing the ground set to a reasonably small set that can be used at test time. The main difference of our framework with the two-stage approaches is that we allow for personalization: A small subset of items that can be found at test time specific to the task at hand. This leads to a completely new problem formulation, and consequently, new algorithms.

2 Problem Statement: Discrete Meta-Learning

Setup. We consider a family of tasks $\mathcal{T} = \{\mathcal{T}_i\}_{i \in I}$, where the set $I$ could be of infinite size. Each task $\mathcal{T}_i$ is represented via a set function $f_i : 2^V \to \mathbb{R}_+$ that measures the reward of a set $S \subseteq V$ for the $i$-th task, and performing the task $\mathcal{T}_i$ would mean to maximize the function $f_i$ subject to a given constraint. For instance, in a recommender system where we aim to recommend a subset of the items to the users, the set $I$ denotes the set of all the possible users and selecting which items to recommend to a user $i \in I$ is viewed as the task $\mathcal{T}_i$. Moreover, the function $f_i$ encodes the users satisfaction, i.e., $f_i(S)$ quantifies how suitable the set of items $S$ is for user $i$. Taking a statistical perspective, we assume that the tasks $\mathcal{T}_i$ occur according to a possibly unknown probability distribution $i \sim p$.

In this paper, we focus on the case where the functions $f_i$ are monotone and submodular set functions and each task $\mathcal{T}_i$ amounts to maximizing $f_i$ under the $k$-cardinality constraint. That is, the task $\mathcal{T}_i$ is to select a subset $S \subseteq V$ of size $k$ such that the value of $f_i(S)$ is maximized. Submodularity of $f_i$ means that for any $A, B \subseteq V$ the following inequality holds $f_i(A) + f_i(B) \geq f_i(A \cup B) + f_i(A \cap B)$. Furthermore, $f_i$ is called monotone if for any $A \subseteq B$ we have $f_i(A) \leq f_i(B)$.

Training and test tasks. We assume access to a collection of training tasks $\{\mathcal{T}_i\}_{i=1}^m$. These are the tasks that we have already experienced, i.e., they correspond to the users that we have already seen. Formally, this means that for each training task $\mathcal{T}_i$, we assume knowledge of the corresponding function $f_i$. In our formulation, each of the training tasks is assumed to be generated i.i.d. according to the distribution $p$. Indeed, eventually we aim to optimize performance at test time, i.e., obtain the best performance for new and unseen tasks generated independently from the distribution $p$. For instance, in our recommendation setting, test tasks correspond to new users that will arrive in the future. Our goal is to use the training tasks to reduce the computation load at test time.

Two extremes of computation. Let us use $\mathcal{T}_{\text{test}}$ (and $f_{\text{test}}$) to denote the task (and its corresponding set function) that we aim to learn at test time. Ideally, if we have sufficient computational power, then we should directly optimize $f_{\text{test}}$ by solving the following problem

$$
\max_{S \in V, |S| \leq k} f_{\text{test}}(S).
$$

(2)
We denote the optimal solution of (2) by $S^{\ast}_{\text{test}}$. For instance, we can use the greedy procedure to solve (2) which leads to a $(1 - 1/e)$-optimal solution using $O(kn)$ evaluations of $f_{\text{test}}$, and through $k$ passes over the ground set. However, the available computational power and time in the test phase is often limited, either because we need to make quick decisions to respond to new users or since we need to save energy. For instance, in real-world advertising or recommendation systems, both these requirements are crucial: many users arrive within each hour which means fast optimization is crucial (especially if $n, k$ are large), and also, reducing computation load would lead to huge energy savings in the long run. In such cases, Problem (2) should be solved approximately with less computation.

An alternative to reduce computation at test time is to solve the problem associated with the expected reward over all possible tasks in the training phase (when we have sufficient computation time), i.e.,

$$\max_{S \in V, |S| \leq k} \mathbb{E}_{i \sim p} [f_i(S)].$$

We denote the optimal solution of (3) by $S^{\ast}_{\text{exp}}$. The rationale behind this approach is that the optimal solution to this problem would generalize well over an unseen task if the new task is also drawn according to the probability distribution $p$. In other words, the solution of (3) should perform well for the problem in (2) that we aim to solve at the test time, assuming that $f_{\text{test}}$ is sampled according to $p$. In this way, we do not need any extra computation at the test time. However, in this case, the solution that we obtain would not be the best possible solution for the task that we observe at the test time, i.e., $S^{\ast}_{\text{test}}$ is not equal to $S^{\ast}_{\text{exp}}$. Note that we often do not have access to the underlying probability distribution $p$, and we only have access to a large set of realizations of tasks in the training phase. As a result, instead of solving (3), we settle for maximizing the sample average function

$$\max_{S \in V, |S| \leq k} \frac{1}{m} \sum_{i=1}^{m} f_i(S),$$

where $m$ is the number of available tasks in the training phase.

Problems (2) and (4) can be considered as two different extreme cases. In the first option, by solving (2), we avoid any pre-processing in the training phase, and we obtain the best possible guarantee for the new task, but at the cost of performing computationally expensive operations (e.g., full greedy) at the test time. In the second approach, by solving (4) in the training phase, we obtain a solution that possibly performs reasonably without any computation at the test phase, but the quality of the solution may not be as good as the first option. In summary, there exists a trade-off between the required computational cost at the test time and the performance guarantee on the unseen task. Hence, a fundamental question that arises is what would be the best scheme at the training phase assuming that at test time we have some limited computational power. For instance, in the monotone submodular case, assume that instead of running the greedy algorithm for $k$ rounds, which has a complexity of $O(kn)$, we can only afford to run $\alpha k$ rounds of greedy at test time, which has a complexity of $O(\alpha nk)$, where $\alpha \in (0, 1)$ is small. In this case, a natural solution would be to find an appropriate set of $(1 - \alpha)k$ elements in the training phase, and add the remaining $\alpha k$ elements at test time when a new task arrives. This discussion also applies to any other greedy method (e.g., lazy or stochastic greedy). We now formally state this problem.

**Discrete Meta-Learning.** As we discussed so far, when computational power is limited at test time, it makes sense to divide the process of choosing the best decision between training and
test phases. To be more specific, in the training phase, we choose a subset of elements from the ground set that would perform over the training tasks, and then select (or optimize) the remaining elements at the test time specifically with respect to the task at hand. To state this problem, consider $S_{tr} \subseteq V$ with cardinality $|S_{tr}| = l$, where $l < k$, as the initial set that we aim to find at the training phase, and the set $S_i$ that we add to the initial set $S_{tr}$ at test time (See Figure 1 for an illustration). Hence, the problem of interest can be written as

$$\max_{S_{tr} \in V, |S_{tr}| \leq l} \mathbb{E}_{i \sim p} \left[ \max_{S_i \in V, |S_i| \leq k-l} f_i(S_{tr} \cup S_i) \right].$$

(5)

Note that the critical decision variable that we need to find is $S_{tr}$ which is the best initial subset of size $l$ overall all possible choices of task when a best subset of size $k - l$ is added to that. In fact, if we define $f_i'(S_{tr}) := \max_{S_i \in V, |S_i| \leq k-l} f_i(S_{tr} \cup S_i)$, then we can rewrite the problem in (5) as

$$\max_{S_{tr} \in V, |S_{tr}| \leq l} \mathbb{E}_{i \sim p} \left[ f_i'(S_{tr}) \right].$$

(6)

As described previously, we often do not have access to the underlying probability distribution $p$ of the tasks, and we instead have access to a large number of sampled tasks that are drawn independently according to $p$. Hence, instead of solving (5), we solve its sample average approximation given by

$$\max_{S_{tr} \in V, |S_{tr}| \leq l} \frac{1}{m} \sum_{i=1}^{m} \left[ \max_{S_i \in V, |S_i| \leq k-l} f_i(S_{tr} \cup S_i) \right] = \max_{S_{tr} \in V, |S_{tr}| \leq l} \frac{1}{m} \sum_{i=1}^{m} f_i'(S_{tr})],$$

(7)

where $m$ is the number of tasks in the training set which are sampled according to $p$. Even though the functions $f_i$ are submodular, $f_i'$ is not submodular or $k$-submodular [OY15] (see Appendix E for specific counter examples). Hence, Problem (7) is not a submodular maximization problem. In the next section, we present algorithms for solving Problem (7) with provable guarantees.

We finally note that Problem (7) will be solved at training time to find the solution $S_{tr}$ of size $l$. This solution is then completed at test time, by, e.g., running $k - l$ further rounds of greedy on the new task, to obtain a task-specific solution of size $k$. 

Figure 1: (a) Optimal sets for each of the training tasks ($k = 6$); (b) the set obtained by solving the average problem in (4); (c) the optimal set for a new task revealed at test time, i.e. solving the problem in (2); (d) the optimal set for the new task is also obtained by solving the meta-learning problem in (7) with $l = 4$ (brown set) and adding the task-specific elements at test time (red set).
3 Algorithms for Discrete Submodular Meta-Learning

Solving Problem (7) requires finding a set $S_{tr}$ for the outer maximization and sets $\{S_i\}_{i=1}^m$ for the inner maximization. In this section, we describe our proposed greedy-type algorithms to select the elements $S_{tr}$ and $\{S_i\}_{i=1}^m$. As we deal with $m + 1$ sets, the order in which the sets $S_{tr}$ and $\{S_i\}_{i=1}^m$ are updated becomes crucial, i.e., it is not clear which of the sets $S_{tr}$ or $S_i$’s should be preferably updated in each round and how can the functions $f_i$ be incorporated in finding the right order, which is the main challenge in designing greedy methods to solve (7). We design greedy procedures with both deterministic and randomized orders and provide strong guarantees for their solutions.

3.1 Deterministic Algorithms

In this section, we first describe Algorithms 1 and 2 which use specific orderings to solve Problem (7). Based on these two, we then design Algorithm 3 as our main deterministic algorithm. Throughout this section, we use $\Delta_i(e|S) = f_i(S \cup \{e\}) - f_i(S)$ to denote the marginal gain of adding an element $e$ to set $S$ for function $f_i$. In brief, Algorithm 1 first fills $S_{tr}$ greedily up to completion and then it constructs each of the $S_i$’s greedily on the top of $S_{tr}$. Specifically, starting from the empty set initialization for $S_{tr}$ and $S_i$’s, Algorithm 1 constructs in its first phase the set $S_{tr}$ in $l$ rounds, by adding one element per round, where the next element in each round is chosen according to $e^* = \arg \max_{e \in V} \Delta_i(e|S_{tr})$. Once $S_{tr}$ is completed, in the second phase, each of the sets $S_i$ is constructed in parallel by running the greedy algorithm on $f_i$. That is, each $S_i$ is updated in $k - l$ rounds where in each round an element with maximum marginal on $f_i$ is added to $S_i$ based on $e^*_i = \arg \max_{e \in V} \Delta_i(e|S_{tr} \cup S_i) - f_i(S_{tr} \cup S_i)$.

Algorithm 2 uses the opposite ordering of Algorithm 1. Initializing with all sets to be empty, in the first phase it constructs the sets $S_i$ using the greedy procedure on $f_i$, i.e., each $S_i$ is updated in parallel in $k - l$ rounds, where in each round the element $e^*_i$ de-
Algorithm 3 Meta-Greedy

1: Run Algorithms 1 and 2 and obtain respective solution sets $S^{(1)}_{tr}, \{S^{(1)}_i\}_{i=1}^m$ and $S^{(2)}_{tr}, \{S^{(2)}_i\}_{i=1}^m$.
2: Compute the objective value $\sum_{i=1}^m f_i(S_{tr} \cup S_i)$ for both solution sets.
3: Return $S_{tr}$ and $S_i$ of the solution set that has a higher objective value.

The solutions obtained by Algorithms 1 and 2 are guaranteed to be near-optimal, it turns out that they can be complementary with respect to each other. Our main deterministic algorithm, called Meta-Greedy, runs both Algorithms 1 and 2 and chooses as output the solution, among the two, that leads to a higher objective value in (7). To be more specific, if we consider $S^{(1)}_{tr}, \{S^{(1)}_i\}_{i=1}^m$ as the outputs of Algorithm 1 and $S^{(2)}_{tr}, \{S^{(2)}_i\}_{i=1}^m$ as the outputs of Algorithm 2, then Meta-Greedy compares the values of $\sum_{i=1}^m f_i(S_{tr} \cup S^{(1)}_i)$ and $\sum_{i=1}^m f_i(S_{tr} \cup S^{(2)}_i)$ and chooses the solution set that has the higher objective function value. Note that as we described earlier, the main output of this procedure should be the set $S_{tr}$ of size $l$. Hence, the output of Meta-Greedy is either $S^{(1)}_{tr}$ or $S^{(2)}_{tr}$ and the sets $\{S^{(1)}_i\}_{i=1}^m$ and $\{S^{(2)}_i\}_{i=1}^m$ are only evaluated for the purpose of comparing objective function values.

Next, we explain why our Meta-Greedy method can outperform both Algorithms 1 and 2. This will be done by providing the theoretical guarantees for these methods and consequently explaining why Algorithms 1 and 2 are complementary.

Theoretical guarantees. We begin with the analysis of Algorithm 1. The following proposition relates the overall performance of Algorithm 1 to its performance after phase 1 and shows that the output of the algorithm is at least 1/2-optimal. We use OPT for the optimal value of Problem (7).

**Proposition 1.** Let $S^{(1)}_{tr}, \{S^{(1)}_i\}_{i=1}^m$ be the output of Algorithm 1, and define $\beta$ as $\beta := \frac{1}{m} \sum_{i=1}^m f_i(S^{(1)}_{tr})$. If the set functions $f_i$ are monotone and submodular, then

$$\frac{1}{m} \sum_{i=1}^m f_i(S^{(1)}_{tr} \cup S^{(1)}_i) \geq \max\{\beta, (1 - 1/e)(OPT - 2\beta) + \beta\}.$$ 

Consequently, the solution obtained by Algorithm 1 is at least 1/2-optimal for any value of $\beta$.

**Proof.** Check Appendix A. ■

The proof of this proposition is relegated to the appendix. The key step in the proof is to relate the progress made in phase 1 to the gap to OPT. This is indeed challenging as phase 1 only involves updates on the outer maximization of (7). In this regard, we prove a novel technical lemma that can be generally applicable to any mini-max submodular problem. The guarantee given in Proposition 1 is minimized when $\beta = OPT/2$. If $\beta$ is small (e.g., $\beta = 0$) or if $\beta$ is large (e.g. if $\beta = (1 - 1/e)OPT$) then the guarantee becomes tight (e.g. $(1 - 1/e)OPT$). This is indeed expected from the greedy nature of the two phases of Algorithm 1. What is non-trivial about the result of Proposition 1 is that it provides a strong guarantee for any
value of $\beta$, and not just cases that $\beta$ is small or large. Similarly, we can provide near-optimality guarantees for Algorithm 2.

**Proposition 2.** Let $S_{tr}^{(2)}$, $\{S_i^{(2)}\}_{i=1}^m$, be the output of Algorithm 2, and define $\gamma$ as $\gamma := \frac{1}{m} \sum_{i=1}^m f_i(S_i^{(2)})$. If the set functions $f_i$ are monotone and submodular, then

$$\frac{1}{m} \sum_{i=1}^m f_i(S_{tr}^{(2)} \cup S_i^{(2)}) \geq \max\{\gamma, (1 - 1/e)(OPT - 2\gamma) + \gamma\}.$$ 

Consequently, the solution obtained by Algorithm 2 is at least $1/2$-optimal for any value of $\gamma$.

**Proof.** Check Appendix B. ■

Similarly, we can show that $\gamma = OPT/2$ leads to (the worst) guarantee $1/2$-OPT, while for large and small values of $\gamma$ the bound in Proposition 2 approaches the optimal approximation $(1 - 1/e)OPT$.

We note that the values $\beta$ in Proposition 1 (Algorithm 1) and $\gamma$ in Proposition 2 (Algorithm 2) represent two different extremes. The value $\beta$ represents the significance of the role of $S_{tr}$ in solving Problem (7), and $\gamma$ represents how significant the role of the sets $\{S_i\}_{i=1}^m$ can be. Even though the worst-case guarantees of Propositions 1 and 2 are obtained when $\beta, \gamma = OPT/2$, a coupled analysis of the algorithms show that in this case at least one of the algorithms should output a solution which is strictly better than $1/2$-optimal. In other words, the outcomes of Algorithms 1 and 2 are dependent to one another, and the best performance is achieved when the maximum of the two is considered. This justifies why our main algorithm Meta-Greedy can perform strictly better than each of the Algorithms 1 and 2. Using a coupled analysis of the outcome of Algorithms 1 and 2, we can bound the performance of Meta-Greedy for different values of $\beta$ and $\gamma$ (see the proof of Theorem 1 in the appendix). In particular, we can show that the output of Meta-Greedy is at least 0.53-optimal. The proof of the following theorem carefully analyzes the interplay between the role of the inner and outer maximization problems in (7). We emphasize that the proof introduces new techniques applicable to other types of minimax submodular problems.

**Theorem 1.** Consider the Meta-Greedy algorithm outlined in Algorithm 3. If the functions $f_i$ are monotone and submodular, then we have

$$\max\left\{\frac{1}{m} \sum_{i=1}^m f_i(S_{tr}^{(1)} \cup S_i^{(1)}), \frac{1}{m} \sum_{i=1}^m f_i(S_{tr}^{(2)} \cup S_i^{(2)})\right\} \geq 0.53 \times OPT.$$ 

**Proof.** Check Appendix C. ■

**Remark 1.** Note that for all the results in Propositions 1 and 2 as well as Theorem 1, for given output sets $S_{tr}$ or $\{S_i\}_{i=1}^m$, the value of $\frac{1}{m} \sum_{i=1}^m f_i(S_{tr} \cup S_i)$ is a lower bound for the objective function value of Problem (7) evaluated at the output set $S_{tr}$. To be more precise, the accurate measure for evaluating the quality of the output set $S_{tr}$ is $\frac{1}{m} \sum_{i=1}^m \left[\max_{S_i \subseteq V, \left|S_i\right| \leq k-1} f_i(S_{tr} \cup S_i)\right]$ which is indeed larger than $\frac{1}{m} \sum_{i=1}^m f_i(S_{tr} \cup S_i)$. Hence, all the guarantees that have obtained in the statements above (as well as Theorem 2 below) would directly translate into the same guarantees when we evaluate the objective in (7) on the set $S_{tr}$.
Let the (random) sets $S_{tr}$ and $\{S_i\}_{i=1}^m$ be the output of Algorithm 4. If the functions $f_i$ are monotone and submodular, then

$$E\left[\frac{1}{m} \sum_{i=1}^m f_i(S_{tr} \cup S_i)\right] \geq \left(1 - \frac{1}{e} - b\right) \text{OPT},$$

where $b \to 0$ as $k - 1$ and $l$ grow. More precisely, letting $c = \max\{\frac{1}{k-1}, \frac{1}{l}\}$, we have $b = c + (\exp(3\sqrt{c\log 1/c}) - 1)/e = O(\sqrt{c\log 1/c})$.

Proof. Check Appendix D. 

Remark 2. All presented algorithms are designed for the training phase and their output is the set $S_{tr}$ with size $l$. The sets $\{S_i\}_{i=1}^m$ are only computed for algorithmic purposes. Given a new task at the test phase, the remaining $k - 1$ task-specific elements will be added to $S_{tr}$ using for instance greedy updates that require a total complexity of $O((k - 1)n)$ in function evaluations. Also, the training complexity of the proposed algorithms is $O(kmn)$, however, certain phases can be implemented in parallel.
4 Simulation Results

We provide two experimental setups to evaluate the performance of our proposed algorithms and compare with other baselines. Each setup involves a different set of tasks which are represented as submodular maximization problems subject to the \( k \)-cardinality constraint. In our experiments, we consider the following algorithms: \textbf{Meta-Greedy} (Algorithm 3), \textbf{Randomized Meta-Greedy} (Algorithm 4), \textbf{Greedy-Train} (which chooses all the \( k \) elements during the training phase–see (4) and the discussion therein), \textbf{Greedy-Test} (which chooses all the \( k \) elements during the test phase–see (2) and the discussion therein), and \textbf{Random} (which chooses a random set of \( k \) elements). In the following, we briefly explain the data and tasks and refer the reader to the supplementary materials for more details.

\textbf{Ride Share Optimization}. We will formalize and solve a facility location problem on the Uber dataset \[\text{Ube}\]. Our experiments were run on the portion of data corresponding to Uber pick-ups in Manhattan in the period of September 2014. This portion consists of \( \sim 10^6 \) data points each represented as a triplet \((\text{latitude}, \text{longitude}, \text{DateTime})\). A customer and a driver are specified through their locations on the map. We use \( u = (x_u, y_u) \) for a customer \( a \) and \( r = (x_r, y_r) \) for a driver. We define the “convenience score” of a (customer, driver) pair as \( c(u, r) = 2 - \frac{2}{1 + e^{-200d(u, r)}} \), where \( d(u, r) \) denotes the Manhattan distance \[\text{MKZK18}\]. Given a specific time \( a \), we define a time slot \( T_a \) and picking inside the data set 10 points in half an hour prior to time \( a \), and for each point we further pick 10 points in its 1 km neighborhood, which makes a total of 100 points (locations) on the map. A task \( T_i \) takes place at a corresponding time \( a_i \), and by defining the set of locations \( T_{ai} \) as above, we let \( f_i \) be a monotone submodular function defined over a set \( S \) of driver locations as \( f_i(S) = \sum_{u \in T_{ai}} \max_{r \in S} c(u, r) \). We pick 100,000 locations at random from the September 2014 Uber pick-up locations as a ground set. For training we form \( m = 50 \) tasks by picking for each task a random time in the first week of Sept. 2014. We test on \( m = 50 \) new tasks formed similarly from the second week of Sept. 2014 and report in the figures the average performance obtained at test tasks.

Figures 2a and 2b show the performance of our proposed algorithms against the baselines.
Figure 3: Performance for Movie Recommendation.

mentioned above. Figure (2a) shows the performance of all algorithms when we fix \( k = 20 \), and vary \( l \) from 5 to 18. Larger \( l \) means less computation at test time (since we need to further choose \( k - l \) elements at test). However, we see that even for large values of \( l \) (e.g., \( l = 16 \)), the performance of Meta-Greedy is still quite close to the ideal performance of Greedy-Test. Putting this together with the fact that the performance of Greedy-Train is not so good, we can conclude that adding a few personalized elements at test time significantly boosts performance to be even close to the ideal. In Figure (2b), we compare the performance of all the algorithms when \( k \) changes from 5 to 30, and \( l \) is 80% of \( k \) (\( l = \lfloor 0.8k \rfloor \)). As we can see, even when we just learn 20% of the set in test time, the performance of Meta-greedy is close to Test-Greedy. Also, when \( k - l \) increases, Random-Meta-Greedy performs better than Meta-Greedy. This is in compliance with the results of Theorems 1, 2.

**Movie Recommendation.** In this application, we use the MovieLens dataset [HK15] which consists of \( 10^6 \) ratings (from 1 to 5) by 6041 users for 4000 movies. We pick the 2000 most rated movies, and 200 users who rated the highest number of movies (similar to [SZKK17]). We partitioned the 200 users into 100 users for the training phase and 100 other users for the test phase. Each movie can belong to one of 18 genres. For each genre \( t \) we let \( G_t \) be the set of all movies with in genre \( t \). For each user \( i \), we let \( R_i \) be the set of all movie rated by the user, and for each movie \( v \in R_i \) the corresponding rating is denoted by \( r_i(v) \). Furthermore, for user \( i \) we define \( f_i(S) = \sum_{t=1}^{18} w_{i,t} \max_{v \in R_i \cap G_t \cap S} r_i(v) \) which is the weighted average over maximum rate that user \( i \) gives to movies from each genre and \( w_{i,t} \) is proportion of movies in genre \( t \) which is rated by user \( i \) out of all the rating he provides. A task \( T_i \) involves 5 users \( i_1, \cdots, i_5 \) and the function assigned to the task is the average of \( f_{i_1}, \cdots, f_{i_5} \). We formed \( m = 50 \) training tasks from the users in the training phase, and \( m = 50 \) test tasks from the users in the test phase. Figure (3a) (resp. 3b) has been obtained in a similar format as Figure 2a (resp. Figure 2b). We observe a very similar pattern as in the ride share experiments.
Comparison with Two-stage Submodular Optimization

Two-stage submodular optimization is another way to deal with limited computational power in test time. In this framework, at training time, a reduced ground set is learned which will be used as a ground set at test time. This procedure will reduce the computational time in test time. More formally, the two-stage submodular optimization framework aims to solve the following problem. Let $f_i : 2^X \to \mathbb{R}_+$ for $i \in [m]$, be a monotone submodular function over ground set $V$. The goal is to find $S$ with size at most $q$ whose subests of size $k$ maximize the sum of $f_i$ for $i \in [m]$:

$$
\max_{S \subseteq X, |S| \leq q} \frac{1}{m} \sum_{i=1}^{m} \max_{S_i \subseteq S, |S_i| \leq k} f_i(S_i)
$$

Once the set $S$ is found, it will be used in the test phase (e.g., by running full greedy on $S$ as the reduced ground set) to find $k$ elements for a new task. This framework uses $O(qk)$ function evaluations for each new test task; however, it poorly personalizes to a test task because the set $S$ has been optimized only for the tasks at the training time. This intuition is indeed consistent with our experimental findings reported below. We further remark that the two-stage framework requires very high computational power in training. For example, the Replacement-Greedy algorithm [SZKK17] requires computational complexity $O(qkmn)$ (which is a factor $q$ larger than the complexity of the algorithms in this paper). As a result of this issue, we were not able to run the state-of-the-art two-stage algorithms to solve (9) in the setting considered in our main simulation results (presented in Section 4). e.g., for ground set of size $n = 10^5$ our two-stage implementation would take a very long time.

We have considered the ride-sharing application discussed in Section 4 and let $n = 500$ (ground set size), $m = 50$ (number of tasks), and $k$ changing from 5 to 30 (cardinality constraint) while $l = 80\%k$ (portion that will fill in the submodular meta-learning during training), and $q = 100$ (size of reduced ground set for two-stage framework). For solving the two-stage problem (9) we have used the Replacement-Greedy algorithm introduced in [SZKK17]. We choose these parameters based on the following two facts:

1. Because of the high computational cost of the Replacement Greedy algorithm in training
for the ride-sharing application, we chose $n$ to be 500.

2. We provide a fair comparison in terms of computational power at test time, which means both Meta-Greedy (our algorithm) and Replacement-Greedy have exactly the same computational cost at test time. Formally, $n(k - l) = qk$.

we report the result for the above setting in the Figure 4. A few comments are in order: (i) The two stage implementation reduces the ground set of size $n = 500$ to $q = 100$. When $k$ is small, some of the popular elements found at training time would be good enough to warrant a good performance at test time. However, when $k$ increases, the role of personalizing becomes more apparent. As we see, the performance of Replacement-Greedy does not improve much when we increase $k$ and it is close to the performance of Greedy-Train (which chooses all the $k$ elements during the training phase—see (4) and the discussion therein). However, since Meta-Greedy does (a small) task-specific optimization at test time, its performance becomes much better. We emphasize again that, in order to be fair, the comparison in Figure 4 has been obtained using the same computational power allowed at test time for both meta-learning and two-stage approaches.

6 Conclusion and Future Work

In this paper, we extended the notion of Model-Agnostic Meta-Learning (MAML) to discrete optimization and in particular to submodular maximization. We proposed a novel formulation in which we aim to find an initial solution set that can be quickly adapted to a new task at a relatively low computational cost. In our meta-learning framework, the process of selecting set items is done in two parts: In the first part, a set of items are selected offline according to prior experience and data. In the second part, which happens at test time, a set of elements that is personalized to the new revealed task is selected. For the proposed problem, we introduced a deterministic variant of the greedy algorithm which obtains a solution that is at least 0.53-optimal, when the tasks are monotone and submodular. We further presented a randomized algorithm that improves this result and obtains $(1 - 1/e - o(1))$-approximation in expectation. We also studied the performance of our proposed meta-learning framework and algorithms for two real-world applications: movie recommendation and ride-sharing problems. Our numerical results indicate the advantage of our proposed scheme with respect to traditional learning procedures as well as methods based on two-stage submodular optimization.

There are numerous open directions that can be investigated along the lines of discrete meta-learning and user-specific adaptation for discrete problems (indeed, this work can be considered as a first step). Examples include extending the results to a more general setting when the tasks are (approximately) submodular but non-monotone, considering the case that the tasks at training and test times are drawn according to two different probability distributions (possibly with bounded distance), and designing

Further, exploring meta-learning continuous extensions and

Acknowledgements

The research of Arman Adibi and Hamed Hassani is supported by NSF award CPS-1837253, NSF CAREER award CIF 1943064, and Air Force Office of Scientific Research Young Investigator Program (AFOSR-YIP) under award FA9550-20-1-0111.
Appendix

A Proof of Proposition 1

Let $S_{tr}$, $\{S_i\}_{i=1}^m$ be the output of Algorithm 1 and $S^*_{tr}$, $\{S^*_i\}_{i=1}^m$ be the optimal solution for problem (7). We first show that the output of Algorithm 1 in phase 1 satisfies the following inequality:

$$
\sum_{i=1}^m f_i(S^*_{tr} \cup S^*_i) - \sum_{i=1}^m f_i(S_{tr}) \leq \sum_{i=1}^m f_i(S_{tr} \cup S^*_i)
$$

To show (10) let $e(t)$ be the $t^{th}$ element of greedy procedure in phase 1, and $S^*_t$ be the $t^{th}$ set in this procedure, where $e(t) = \arg\max_e \sum_{i=1}^m f_i(S^*_{tr} \cup S^*_i) - f_i(S_{tr})$. Let $J(0) = S^*_{tr}$ and define $J(t)$ iteratively as follows. Let $D(t) = J(t-1) \setminus S^*_{tr}$ and define $o(t)$ in the following way:

1. If $e(t) \in D(t)$, then let $o(t) = e(t)$.
2. Otherwise, if $e(t) \notin D(t)$, let $o(t)$ be one of the elements of $D^t$ chosen uniformly at random.

Define $J(t) := J(t-1) \cup e(t) \setminus o(t)$. We show this procedure in the following chain.

$$(S^*_{tr}, \{S^*_i\}_{i=1}^m) \xrightarrow{\{e(t)\}_{t=1}^m} (J(1), \{S^*_i\}_{i=1}^m) \to (J(t), \{S^*_i\}_{i=1}^m)$$

then we can write the following inequalities:

$$
\sum_{i=1}^m f_i(S^*_{tr} \cup S^*_i) - f_i(S_{tr}) = \sum_{i=1}^m f_i(S^*_{tr} \cup e(t)) - f_i(S_{tr} \cup e(t))
$$

$$
\geq \sum_{i=1}^m f_i(S^*_{tr} \cup o(t)) - f_i(S_{tr} \cup o(t))
$$

$$
\geq \sum_{i=1}^m f_i(S^*_i \cup J(t-1)) - f_i(S^*_i \cup J(t-1) \setminus o(t))
$$

$$
\geq \sum_{i=1}^m f_i(S^*_i \cup J(t-1)) - f_i(S^*_i \cup J(t-1) \setminus o(t)) + \sum_{i=1}^m f_i(S^*_i \cup J(t)) + f_i(S^*_i \cup J(t-1) \setminus o(t))
$$

$$
= \sum_{i=1}^m f_i(S^*_i \cup J(t-1)) - f_i(S^*_i \cup J(t))
$$

where (12) follows from definition of $e(t)$ and the greedy procedure and (13) follows from submodularity since in each step $S^*_{tr} \subseteq J(t-1)$ and $o(t) \notin S^*_{tr}$ and finally, equation (14)
follows from the fact that \(-f_i(J^{(t)} \cup S_i^*) + f_i(J^{(t-1)} \cup S_i^* \setminus o^{(t)}) \leq 0\). Then, by summing over \(t\) from 0 to \(l\) we get the following inequality:

\[
\sum_{i=1}^{m} f_i(S_{tr}) = \sum_{i=1}^{m} f_i(S_{tr}^{(l)}) - f_i(S_{tr}^{(0)}) = \sum_{i=1}^{m} \sum_{t=0}^{l} f_i(S_{tr}^{(t)}) - f_i(S_{tr}^{(t-1)}) \\
\geq \sum_{i=1}^{m} \sum_{t=0}^{l} f_i(S_{tr}^{*} \cup J^{(t-1)}) - f_i(S_{tr}^{*} \cup J^{(t)}) \\
= \sum_{i=1}^{m} f_i(S_{tr}^{*} \cup J^{(0)}) - f_i(S_{tr}^{*} \cup J^{(l)}) \\
= \sum_{i=1}^{m} f_i(S_{tr}^{*} \cup S_{tr}^{*}) - f_i(S_{tr}^{*} \cup S_{tr}) \\
= \sum_{i=1}^{m} f_i(S_{tr}^{*} \cup S_{tr}^{*}) - f_i(S_{tr}^{*} \cup S_{tr})
\]

where the last equality comes from the process of defining \(J\). Because, we only change one element by adding element found in greedy process and removing one element from the optimal set in each step and the size of \(J^{(l)}\) is \(l\) in each step; therefore, after \(l\) step \(J^{(l)} = S_{tr}\). By rearranging the terms and summing over \(i\) the claim in (10) follows.

Second, for the phase 2 of the algorithm 1 we can use the usual analysis of greedy [KG] for set \(S_i\):

\[
\sum_{i=1}^{m} f_i(S_{tr} \cup S_i) - f_i(S_{tr}) \geq (1 - \frac{1}{e}) \left( \sum_{i=1}^{m} f_i(S_{tr} \cup S_{tr}^{opt}) - f_i(S_{tr}) \right) \\
\geq (1 - \frac{1}{e}) \left( \sum_{i=1}^{m} f_i(S_{tr} \cup S_{tr}^{*}) - f_i(S_{tr}) \right) \\
\geq (1 - \frac{1}{e}) \left( \sum_{i=1}^{m} f_i(S_{tr}^{*} \cup S_{tr}^{*}) - 2f_i(S_{tr}) \right)
\]

where \(S_{tr}^{opt} = \arg \max_{|S_i| \leq k-l} f_i(S_{tr} \cup S_i)\) in the equation (20). Equation (20) follows from usual greedy analysis, equation (21) follows from definition of \(S_{tr}^{opt}\), and equation (22) follows from equation (10).

Now divide both sides of (10) by \(1/m\) and regroup the terms to obtain

\[
\frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr} \cup S_i) \geq \left( 1 - \frac{1}{e} \right) (\text{OPT} - 2\beta) + \beta,
\]

where \(\beta := \frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr})\).

Finally, since \(S_i \subseteq S_i \cup S_{tr}\) by monotonicity \(f_i(S_i \cup S_{tr}) \geq f_i(S_{tr})\). Then, combing this observation with the result in (23) implies

\[
\frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr} \cup S_i) \geq \max \left\{ \beta, (1 - 1/e)(\text{OPT} - 2\beta) + \beta \right\}.
\]
B Proof of Proposition 2

Let $S_{tr}$, $\{S_i\}_{i=1}^m$ be the output of Algorithm 2 and $S^*_i$, $\{S^*_i\}_{i=1}^m$ be the optimal solution for problem (7). We first show the following about the output of algorithm 2, phase 1.

$$
\sum_{i=1}^m f_i(S^*_i \cup S^*_i) - \sum_{i=1}^m f_i(S_i) \leq \sum_{i=1}^m f_i(S^*_i \cup S_i)
$$

(24)

to show (24) consider the following:

let $e_i^{(t)} = \arg\max_e f_i(S_i^{(t-1)} \cup e) - f_i(S_i^{(t-1)})$. let $f_i^{(0)} = S^*_i$ and define $f_i^{(t)}$ iteratively as follows. Let $D_i^t = J_i^{(t-1)} \setminus S_i^{(t-1)}$ and define $o_i^{(t)}$ in the following way:

1. If $e_i^{(t)} \in D_i^t$, then $o_i^{(t)} = e_i^{(t)}$;
2. Otherwise, if $e_i^{(t)} \notin D_i^t$, let $o_i^{(t)}$ be one of the elements of $D_i^t$ chosen uniformly at random;

Define $J_i^{(t)} := J_i^{(t-1)} \cup e_i^{(t)} \setminus o_i^{(t)}$.

$$
(S_{tr}, \{S_i\}_{i=1}^m) \xrightarrow{\{e_i^{(t)}\}} (S_{tr}^*, \{J_i^{(1)}\}_{i=1}^m) \ldots \xrightarrow{\{e_i^{(k-1)}\}} (S_{tr}^*, \{J_i^{(k-1)}\}_{i=1}^m)
$$

then we can write the following inequalities:

$$
f_i(S_i^{(t)}) - f_i(S_i^{(t-1)}) = f_i(S_i^{(t-1)} \cup e_i^{(t)}) - f_i(S_i^{(t-1)})
$$

(25)

$$
\geq f_i(S_i^{(t-1)} \cup o_i^{(t)}) - f_i(S_i^{(t-1)})
$$

(26)

$$
\geq f_i(S_{tr}^* \cup J_i^{(t-1)}) - f_i(S_{tr}^* \cup J_i^{(t-1)} \setminus o_i^{(t)})
$$

(27)

$$
\geq f_i(S_{tr}^* \cup J_i^{(t-1)}) - f_i(S_{tr}^* \cup J_i^{(t-1)} \setminus o_i^{(t)}) - f_i(S_{tr}^* \cup J_i^{(t)}) + f_i(S_{tr}^* \cup J_i^{(t-1)} \setminus o_i^{(t)})
$$

(28)

$$
= f_i(S_{tr}^* \cup J_i^{(t-1)}) - f_i(S_{tr}^* \cup J_i^{(t)})
$$

(29)

where (26) follows from definition of $e_i^{(t)}$ and the greedy procedure and (27) follows from the submodularity since in each step $S_i^{(t-1)} \subseteq J_i^{(t-1)}$ and $o_i^{(t)} \notin S_i^{(t-1)}$ and finally, equation (28) follows from the fact that $-f_i(S_{tr}^* \cup J_i^{(t-1)}) + f_i(S_{tr}^* \cup J_i^{(t-1)} \setminus o_i^{(t)}) \leq 0$ because of monotonicity. Then, by summing over $t$ from 0 to $k - l$ we get the following inequality:

$$
f_i(S_i) = f_i(S_i^{(k-1)}) - f_i(S_i^{(0)}) = \sum_{t=0}^{k-l} f_i(S_i^{(t)}) - f_i(S_i^{(t-1)})
$$

(30)

$$
\geq \sum_{t=0}^{k-l} f_i(S_{tr}^* \cup J_i^{(t-1)}) - f_i(S_{tr}^* \cup J_i^{(t)})
$$

(31)

$$
= f_i(S_{tr}^* \cup J_i^{(0)}) - f_i(S_{tr}^* \cup J_i^{(k-1)})
$$

(32)

$$
= f_i(S_{tr}^* \cup S_i^*) - f_i(S_{tr}^* \cup S_i)
$$

(33)
where the last equality comes from the process of defining $f_i^{(k-l)}$; since, the size of $f_i^{(l)}$ is $k-l$ in each step and after $k-l$ step $f_i^{(k-l)} = S_i$. Then, by rearranging and summing over $i$ we can obtain (24).

Second, for phase 2 of algorithm 2 we can use the usual analysis of greedy [KG] for set $S_{tr}$:

$$
\sum_{i=1}^{m} f_i(S_{tr} \cup S_i) - f_i(S_i) \geq (1 - \frac{1}{e}) \left( \sum_{i=1}^{m} f_i^{opt}(S_{tr} \cup S_i) - f_i(S_i) \right) 
$$

(34)

\[ \geq (1 - \frac{1}{e}) \left( \sum_{i=1}^{m} f_i^{*}(S_{tr} \cup S_i) - f_i(S_i) \right) \]  

(35)

\[ \geq (1 - \frac{1}{e}) \left( \sum_{i=1}^{m} f_i^{*}(S_{tr} \cup S_i^{*}) - 2f_i(S_i) \right) \]  

(36)

where $S_{tr}^{opt} = \arg \max_{|S_{tr}| \leq l} \sum_{i=1}^{m} f_i(S_{tr} \cup S_i)$ in equation (34). Equation (34) follows from the usual greedy analysis, equation (35) follows from the definition of $S_{tr}^{opt}$, and equation (36) follows from equation (24).

Now divide both sides of (36) by $1/m$ and regroup the terms to obtain

$$
\frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr} \cup S_i) \geq \left( 1 - \frac{1}{e} \right) (\text{OPT} - 2\gamma) + \gamma,
$$

(37)

where $\gamma := \frac{1}{m} \sum_{i=1}^{m} f_i(S_i)$.

Finally, since $S_i \subseteq S_i \cup S_{tr}$ by monotonicity $f_i(S_i \cup S_{tr}) \geq f_i(S_i)$. Then, by combing this result wit (37) we obtain

$$
\frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr} \cup S_i) \geq \max \left\{ \gamma, (1 - 1/e)(\text{OPT} - 2\gamma) + \gamma \right\}.
$$

The following shows the ratio of lower bound to optimum (a similar plot can be obtained for the lower bound of Proposition 1 when $\gamma$ is replaced with $\beta$). As we observe, in the worst case, the approximation factor is 0.5.

---

**Figure 5:** y-axis: The lower bound of Proposition 2 divided by OPT, x-axis: $\gamma$/OPT.
C Proof of Theorem 1

Let $\theta_2 = \frac{1}{m} \sum_{i=1}^{m} f_i(S_i^{(2)} \cup S_i^*)$. Since $S_i^{(2)}$ found greedily given $\{S_i\}_{i=1}^{m}$ we can write:

$$\theta_2 - \gamma \geq (\text{OPT} - \gamma)(1 - \frac{1}{e}) \geq \left(\frac{1}{m} \sum_{i=1}^{m} f_i(S' \cup S_i^{(2)}) - \gamma\right)(1 - \frac{1}{e})$$  \hspace{1cm} (38)

for every $|S'| \leq l$. Also, we can write

$$\text{OPT} - \gamma = \frac{1}{m} \sum_{i=1}^{m} f_i(S_i^* \cup S_i^*) - f_i(S_i^{(2)})$$  \hspace{1cm} (39)

$$\leq \frac{1}{m} \sum_{i=1}^{m} f_i(S_i^* \cup S_i^{(2)} \cup S_i^*) - f_i(S_i^{(2)})$$  \hspace{1cm} (40)

$$= \frac{1}{m} \sum_{i=1}^{m} f_i(S_i^* \cup S_i^{(2)} \cup S_i^*) + f_i(S_i^* \cup S_i^{(2)}) - f_i(S_i^* \cup S_i^{(2)}) - f_i(S_i^{(2)})$$  \hspace{1cm} (41)

$$\leq \frac{1}{m} \sum_{i=1}^{m} f_i(S_i^* \cup S_i^{(2)} \cup S_i^*) - f_i(S_i^* \cup S_i^{(2)}) + \frac{\theta_2 - \gamma}{1 - 1/e}$$  \hspace{1cm} (42)

$$\leq \frac{1}{m} \sum_{i=1}^{m} f_i(S_i^{(2)} \cup S_i^*) - f_i(S_i^{(2)}) + \frac{\theta_2 - \gamma}{1 - 1/e}$$  \hspace{1cm} (43)

where (42) comes from (38), and (43) comes from submodularity. We thus obtain

$$\text{OPT} - \frac{\theta_2 - \gamma}{1 - 1/e} - \gamma \leq \frac{1}{m} \sum_{i=1}^{m} f_i(S_i^{(2)} \cup S_i^*) - f_i(S_i^{(2)})$$  \hspace{1cm} (44)

Also we can write for any set $S'$ such that $|S'| \leq l$:

$$\frac{1}{m} \sum_{i=1}^{m} f_i(S' \cup S_i^*) - f_i(S') \geq \frac{1}{m} \sum_{i=1}^{m} f_i(S' \cup S_i^* \cup S_i) - f_i(S' \cup S_i)$$  \hspace{1cm} (45)

$$\geq \frac{1}{m} \sum_{i=1}^{m} f_i(S' \cup S_i^* \cup S_i) - f_i(S_i) + f_i(S_i) - f_i(S' \cup S_i)$$  \hspace{1cm} (46)

$$\geq \frac{1}{m} \sum_{i=1}^{m} f_i(S_i^* \cup S_i) - f_i(S_i) + f_i(S_i) - f_i(S' \cup S_i)$$  \hspace{1cm} (47)

$$\geq \text{OPT} - \frac{\theta_2 - \gamma}{1 - 1/e} - \gamma + \frac{1}{m} \sum_{i=1}^{m} f_i(S_i) - f_i(S' \cup S_i)$$  \hspace{1cm} (48)

$$\geq \text{OPT} - 2\frac{\theta_2 - \gamma}{1 - 1/e} - \gamma$$  \hspace{1cm} (49)

where (45) follows from submodularity, (47) follows from monotonicity, and (48) follows from (44), and (49) follows from (38). This results the following for any set $S'$ such that $|S'| \leq l$:

$$\frac{1}{m} \sum_{i=1}^{m} f_i(S' \cup S_i^*) - f_i(S') \geq \text{OPT} - 2\frac{\theta_2 - \gamma}{1 - 1/e} - \gamma$$  \hspace{1cm} (50)
Now, from (50) we can find a new bound for the performance of algorithm 3. From (50) we can write:

\[ \frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr}^{(1)} \cup S_i^{(1)}) - f_i(S_{tr}^{(1)}) \geq \text{OPT} - 2 \frac{\theta_2 - \gamma}{1 - 1/e} - \gamma \]  

(51)

Also, since in Algorithm 1 the set \( S_i^{(1)} \) is constructed greedily on the top of \( S_{tr}^{(1)} \), we have:

\[ \frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr}^{(1)} \cup S_i^{(1)}) - \beta \geq \left( \frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr}^{(1)} \cup S_i^{(1)}) - \beta \right)(1 - 1/e) \]

(52)

\[ \geq (\text{OPT} - 2 \frac{\theta_2 - \gamma}{1 - 1/e} - \gamma)(1 - 1/e), \]  

(53)

where (53) follows from (51). We thus obtain:

\[ \frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr}^{(1)} \cup S_i^{(1)}) \geq (\text{OPT} - 2 \frac{\theta_2 - \gamma}{1 - 1/e} - \gamma)(1 - 1/e) + \beta \]  

(54)

Using the same procedure as above, by defining \( \theta_1 = \frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr}^{(1)} \cup S_i^{(1)}) \), we can prove:

\[ \frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr}^{(2)} \cup S_i^{(2)}) \geq (\text{OPT} - 2 \frac{\theta_1 - \gamma}{1 - 1/e} - \gamma)(1 - 1/e) + \beta \]  

(55)

which results in the following lower bound:

\[ \max \left\{ \frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr}^{(1)} \cup S_i^{(1)}) , \frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr}^{(2)} \cup S_i^{(2)}) \right\} \]

\[ \geq \max \left\{ \theta_1, \theta_2, (1 - 1/e)(\text{OPT} - \gamma) + \beta - 2(\theta_2 - \gamma), (1 - 1/e)(\text{OPT} - \beta) + \gamma - 2(\theta_1 - \beta) \right\}. \]  

(56)

Finally, given (54) and (56), the factor 0.53 is obtained as a result of the following procedure. Let \( \beta \) and \( \gamma \) given as \( \beta := \frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr}^{(1)}) \) and \( \gamma := \frac{1}{m} \sum_{i=1}^{m} f_i(S_i^{(2)}) \). Then the left-hand-side term in (8) is lower bounded by:

\[ \min_{\theta_1, \theta_2} \max \left\{ \theta_1, \theta_2, (1 - 1/e)(\text{OPT} - \gamma) + \beta - 2(\theta_2 - \gamma), (1 - 1/e)(\text{OPT} - \beta) + \gamma - 2(\theta_1 - \beta) \right\} \]

subject to \( \theta_1 \geq \max \{ \beta, (1 - 1/e)(\text{OPT} - 2\beta) + \beta \} \)

\( \theta_2 \geq \max \{ \gamma, (1 - 1/e)(\text{OPT} - 2\gamma) + \gamma \} \)

Note that the constraints hold due to the results of Proposition 1 and 2. In particular, the above bound is always larger than 0.53 \times \text{OPT} for any value of \( \beta \) and \( \gamma \).
D Proof of Theorem 2

Consider round \( t \) in which \(| S_{tr} | < l \) and \(| S_i | < k - l \) the expected gain of the algorithm with probability \( \frac{l}{k} \) is the maximum gain from adding an element \( e^* = \arg \max e \sum_{i=1}^{m} f_i(S_{tr}^i \cup e \cup S_i^i) - f_i(S_{tr}^i \cup S_i^i) \) or with probability \( \frac{k-l}{k} \) the gain is \( \sum_{i=1}^{m} \max_{e_i} f_i(S_{tr}^i \cup e_i \cup S_i^i) - f_i(S_{tr}^i \cup S_i^i) \) which can be written as follows.

\[
\mathbb{E}[\sum_{i=1}^{m} f_i(S_{tr}^{i+1} \cup S_i^{i+1}) - f_i(S_{tr}^i \cup S_i^i)|S_{tr},S_i] = \frac{l}{k} \max_{e} \sum_{i=1}^{m} f_i(S_{tr}^i \cup e \cup S_i^i) - f_i(S_{tr}^i \cup S_i^i) + \frac{k-l}{k} \sum_{i=1}^{m} \max_{e_i} f_i(S_{tr}^i \cup e_i \cup S_i^i) - f_i(S_{tr}^i \cup S_i^i)
\]  \hspace{1cm} (58)

assuming \( S_{tr}^i, S_i^i \) is optimal solution, we can also write:

\[
\frac{1}{k} \sum_{i=1}^{m} f_i(S_{tr}^i \cup S_i^i) - f_i(S_{tr}^i \cup S_i^i) \leq \frac{1}{k} \sum_{i=1}^{m} f_i(S_{tr}^i \cup S_i^i \cup S_{tr}^i \cup S_i^i) - f_i(S_{tr}^i \cup S_i^i)
\]  \hspace{1cm} (59)

\[
\leq \frac{1}{k} \sum_{e \in S_{tr}^i \setminus S_i^i} \sum_{i=1}^{m} f_i(e \cup S_{tr}^i \cup S_i^i) - f_i(S_{tr}^i \cup S_i^i)
\]

\[
+ \frac{1}{k} \sum_{i=1}^{m} \sum_{e \in S_{tr}^i \setminus S_i^i} f_i(e \cup S_{tr}^i \cup S_i^i) - f_i(S_{tr}^i \cup S_i^i)
\]  \hspace{1cm} (60)

\[
\leq \frac{l}{k} \max_{e} \sum_{i=1}^{m} f_i(S_{tr}^i \cup e \cup S_i^i) - f_i(S_{tr}^i \cup S_i^i)
\]

\[
+ \frac{k-l}{k} \sum_{i=1}^{m} \max_{e_i} f_i(S_{tr}^i \cup e_i \cup S_i^i) - f_i(S_{tr}^i \cup S_i^i)
\]  \hspace{1cm} (61)

where (59) follows from monotonicity, and (60) follows from submodularity. Then, from (61) and (58) we conclude that:

\[
\mathbb{E}[\sum_{i=1}^{m} f_i(S_{tr}^{i+1} \cup S_i^{i+1}) - f_i(S_{tr}^i \cup S_i^i)|S_{tr},S_i] \leq \frac{1}{k} \sum_{i=1}^{m} f_i(S_{tr}^i \cup S_i^i) - f_i(S_{tr}^i \cup S_i^i)
\]  \hspace{1cm} (62)

In other words, the expected improvement in the objective (left-hand side of (62)) is at least \( 1/k \) times the gap of the current objective value to OPT (i.e. right-hand side of (62)). Note that (62) is only valid when \(| S_{tr} | < l \) and \(| S_i | < k - l \). Hence, by defining the stopping time \( \tau \) as first time that either \(| S_{tr} | = l \) or \(| S_i | = k - l \), and a telescopic usages of the bounds in (62), we obtain the following bound:

\[
\mathbb{E}\left[ \frac{1}{m} \sum_{i=1}^{m} f_i(S_{tr}^i \cup S_i^i) \right] \geq \text{OPT} \times \mathbb{E}\left[ 1 - \left( 1 - \frac{1}{k} \right)^\tau \right]
\]

The following theorem finds an upper bound on \( \mathbb{E}[1 - (1 - \frac{1}{k})^\tau] \) which finishes the proof.

**Lemma 1.** If stopping time \( \tau \) is first time that either \(| S_{tr} | = l \) or \(| S_i | = k - l \) then \( \mathbb{E}[1 - (1 - \frac{1}{k})^\tau] \leq c + \exp(-1 + \sqrt{3c \log(\frac{1}{\epsilon})}) \) where \( c = \min{\|l-I\|} \).
Proof. let \( u_1, u_2, \ldots \) be i.i.d random variables with distribution \( u_i \sim \text{Bernoulli}(1 - l/k) \), i.e. \( p(u_i = 1) = (k - l)/k \). The stopping time \( \tau \) is the first time that \( \sum_{i=1}^{\tau} u_i = k - l \) or \( \tau - \sum_{i=1}^{\tau} u_i = l \). Let us define \( X_r = \sum_{i=1}^{\tau} u_i \).

Furthermore, we define \( \tau' = r \) when \( r \) is the first time that \( X_r = r - l \) and \( \tau'' = r \) when \( r \) is the first time that \( X_r = k - l \). Also, let \( c = \frac{1}{\min\{l, k-l\}} \) as it was defined in the lemma. By this definition, \( \tau = \min\{\tau'', \tau'\} \) and we can write the following about the probabilities of \( \tau' \) and \( \tau'' \):

\[
p(\tau' = r) = \binom{r-1}{l-1} \left( \frac{k-l}{k} \right)^{r-l} \left( \frac{l}{k} \right)^l
\]

\[
p(\tau'' = r) = \binom{r-1}{k-l-1} \left( \frac{l}{k} \right)^{r-k+l} \left( \frac{k-l}{k} \right)^{k-l}
\]

then, based on the definition of \( \tau' \) and \( \tau'' \) we have the following properties for \( \tau' \) and \( \tau'' \):

- if \( r < k - l \) then \( p(\tau'' = r) = 0 \).
- if \( r < l \) then \( p(\tau' = r) = 0 \).
- if \( r > k \) then \( p(\tau' \leq \tau'' \mid \tau' = r) = 0 \).
- if \( r < k \) then \( p(\tau' \leq \tau'' \mid \tau' = r) = 1 \).
- if \( r < k \) then \( p(\tau' \geq \tau'' \mid \tau' = r) = 1 \).
- if \( r > k \) then \( p(\tau' \geq \tau'' \mid \tau' = r) = 0 \).
- \( p(\tau'' = r \mid \tau' \geq \tau'') = p(\tau = r \mid \tau' \geq \tau'') \).
- \( p(\tau' = r \mid \tau' \leq \tau'') = p(\tau = r \mid \tau' \leq \tau'') \).

Moreover using Bayes rule we can write:

\[
p(\tau' = r \mid \tau' \leq \tau'') = \frac{p(\tau' \leq \tau'' \mid \tau' = r) p(\tau' = r)}{p(\tau' \leq \tau'')}
\]

\[
p(\tau' = r \mid \tau' \geq \tau'') = \frac{p(\tau'' \leq \tau' \mid \tau' = r)}{p(\tau'' \leq \tau'')}
\]

Let \( \bar{X}_r = r - X_r \) we can write \( \bar{X}_r = \sum_{i=1}^{r} v_i \) where \( v_1, v_2, v_3, \ldots \) are i.i.d random variable with distribution \( v_i \sim \text{Bernoulli}(l/k) \). Then, we can write the following using Chernoff bound:

\[
p(\tau' = r) \leq p(X_r = r - l) \leq p(X_r \geq l)
\]

\[
\leq p(\bar{X}_r \geq r \left( \frac{l}{k} \right) - (k - r) \left( \frac{l}{k} \right)) \leq \exp \left( -\frac{(k - r)^2 (l)}{3r \left( \frac{l}{k} \right)} \right)
\]

\[
= \exp \left( -\frac{(k - r)^2 (l)}{3rk} \right)
\]
Similarly:

\[ p(\tau'' = r) \leq p(X_r = k - l) \]
\[ \leq p(X_r \geq k - l) \]
\[ \leq p(X_r \geq r(1 - \frac{1}{k}) - (k - r)(1 - \frac{1}{k})) \]
\[ \leq \exp \left( \frac{(k - r)^2(1 - \frac{1}{k})^2}{3r(1 - \frac{1}{k})} \right) \]
\[ \leq \exp \left( \frac{(k - r)^2(k - l)}{3rk} \right) \]
\[ \leq \exp \left( \frac{(k - r)^2}{3rc} \right) \]

then we can write the \( \mathbb{E}[(1 - \frac{1}{k})^\tau] \) as follows:

\[ \mathbb{E}[(1 - \frac{1}{k})^\tau] = \sum_{r=1}^{k} (1 - \frac{1}{k})^r p(\tau = r) \leq (1 - \frac{1}{k})^{k - \alpha \sqrt{c}} \sum_{r=1}^{k} (1 - \frac{1}{k})^r p(\tau = r) \] (74)

Our goal is to find proper bound for (74). We focus on the second term in (75)-(81) and try to find proper bound for it.

\[ \sum_{r=1}^{k} (1 - \frac{1}{k})^r p(\tau = r) \]
\[ = \sum_{r=1}^{k - \alpha \sqrt{c}} (1 - \frac{1}{k})^r (p(\tau' = r) + p(\tau'' = r)) \]
\[ = \sum_{r=1}^{k - \alpha \sqrt{c}} (1 - \frac{1}{k})^r (p(\tau' = r) + p(\tau'' = r)) \]
\[ = \sum_{r=1}^{k - \alpha \sqrt{c}} (1 - \frac{1}{k})^r p(\tau' = r) + \sum_{r=k-l}^{k} (1 - \frac{1}{k})^r p(\tau'' = r) \]
\[ \leq \sum_{r=1}^{k - \alpha \sqrt{c}} \exp \left( \frac{-(k-r)^2}{3rk} \right) + \sum_{r=k-l}^{k} \exp \left( \frac{-(k-r)^2}{3rk} \right) \]
\[ \leq (k - l) \exp \left( \frac{-(k - (k - \alpha \sqrt{c}))^2}{3k^2c} \right) + l \exp \left( \frac{-(k - (k - \alpha \sqrt{c}))^2}{3k^2} \right) \]
\[ \leq (k - l) \exp \left( \frac{-(\alpha \sqrt{c})^2}{3k^2c} \right) + l \exp \left( \frac{-(\alpha \sqrt{c})^2}{3k^2} \right) \]

where (76) follows from law of total probability, (77) follows from bayes rule, (79) follows from Chernoff bound, (80) follows from the fact that \( r < k \). Let \( \alpha = 3\sqrt{\log(\frac{1}{\varepsilon})}.k \). As result, we have:

\[ \sum_{r=1}^{k - \alpha \sqrt{c}} (1 - \frac{1}{k})^r p(\tau = r) \leq (k - l)c^3 + lc^{3l} \] (82)
Assume without loss of generality \( k - l \leq 1 \) and \( k - l \geq 2 \). As a result, \( c = \frac{1}{k - l} \). we want to show that \((k - l)c^3 + l^3cl = c^2 + l^3cl \leq c \). To show this, we show the following equivalent inequality:

\[
I(k - l)^{3cl} \leq c(1 - c) = \frac{k - l - 1}{(k - l)^2}
\]

This holds since \( k - l \geq 2 \) we have \( \frac{I}{(k - l)^3} (k - l)^{-3(c - 1)} \leq \frac{I}{(k - l)^3} 2^{-3(c - 1)} \leq \frac{I}{(k - l)^3} \frac{1}{(k - l)^2} \leq \frac{k - l - 1}{(k - l)^2} \). Moreover, we can bound the first term in (74) as follows:

\[
(1 - \frac{1}{k})^{k - \alpha \sqrt{c}} \leq \exp(-1 + 3\sqrt{c. \log(\frac{1}{c})})
\]

summing up we can find the following bound for \( \mathbb{E}[(1 - \frac{1}{k})^\tau] \) which finishes the proof.

\[
\mathbb{E}[(1 - \frac{1}{k})^\tau] \leq c + \exp(-1 + 3\sqrt{c. \log(\frac{1}{c})})
\]

\[\blacksquare\]

E Counter-example for Submodularity of the Objective in (7)

In this section, we provide a counterexample for submodularity of the objective function in the equation (7). We consider a maximum coverage problem in which the function value is an area covered by a set of elements. We define the ground set \( V = \{ABIJ, BCDI, ACDJ, IDEH, HEFG, BCEH\} \) which has shown in Figure 6. Each element is a rectangle, and a function value of that element is an area covered by that element. We refer to each element (rectangle) by it’s vertices.

Figure 6: Counter Example of Submodularity
Let $AC = CD = DE = EF = 1$, and $BC = 0.75$. Also in (7) we let $m = 1$ and $k - l = 1$ which means that we are considering a single set function $f$ defined as: $f(S) = \max_{e \in V} A(S \cup e)$, where $A(T)$ is a area of set $T$. Note that the area function $A$ is monotone and submodular, however as we will show below, the function $f$ is not submodular. To do so, we consider two sets $T_1 = \emptyset$ and $T_2 = \{ACDJ\}$ and add the element IDEH to both sets and observe that $f$ does not satisfy the diminishing returns property. Let us first compute the function value at $T_1$ and $T_2$ as follows:

$$f(T_1) = \max_{e \in V} A(e) = A(\{BCEH\}) = 1.5,$$

and

$$f(T_2) = \max_{e \in V} A(T_2 \cup e) = A(\{ACDJ, IDEH\}) = 1.75.$$

Similarly, we compute the function value at $T'_1 = T_1 \cup \{IDEH\}$, and $T'_2 = T_2 \cup \{IDEH\}$:

$$f(T'_1) = \max_{e \in V} A(T'_1 \cup e) = A(\{IDEH, ACDJ\}) = 1.75,$$

and

$$f(T'_2) = \max_{e \in V} A(T'_2 \cup e) = A(\{IDEH, ACDJ, EFGH\}) = 2.5.$$

We can now see that $T_1 \subseteq T_2$, but $f(T'_2) - f(T_2) \not\leq f(T'_1) - f(T_1)$. Therefore, $f$ is not submodular.

Also let us make a remark about $k$-submodularity which studies functions of $k$ subsets of the ground set that are disjoint sets. This class of functions is submodular in each orthant [OY15]. However, in the submodular meta-learning framework, sets can have overlap, and there is no restriction on the sets to be disjoint. Therefore, our framework is different from $k$-submodular maximization.
References

[AES19] Antreas Antoniou, Harrison Edwards, and Amos J. Storkey. How to train your MAML. In 7th International Conference on Learning Representations, ICLR, 2019.

[BBC90] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. Learning a synaptic learning rule. Citeseer, 1990.

[BENW15] Rafael Barbosa, Alina Ene, Huy Nguyen, and Justin Ward. The power of randomization: Distributed submodular maximization on massive datasets. In International Conference on Machine Learning, pages 1236–1244, 2015.

[BMKS16] Eric Balkanski, Baharan Mirzasoleiman, Andreas Krause, and Yaron Singer. Learning sparse combinatorial representations via two-stage submodular maximization. In International Conference on Machine Learning, pages 2207–2216, 2016.

[BRS19] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. An exponential speedup in parallel running time for submodular maximization without loss in approximation. In Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 283–302. SIAM, 2019.

[CHHK18] Lin Chen, Christopher Harshaw, Hamed Hassani, and Amin Karbasi. Projection-free online optimization with stochastic gradient: From convexity to submodularity. arXiv preprint arXiv:1802.08183, 2018.

[CMS20] Liam Collins, Aryan Mokhtari, and Sanjay Shakkottai. Distribution-agnostic model-agnostic meta-learning. arXiv preprint arXiv:2002.04766, 2020.

[DK11] Abhimanyu Das and David Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. arXiv preprint arXiv:1102.3975, 2011.

[DSC+16] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL$^2$: Fast reinforcement learning via slow reinforcement learning. CoRR, abs/1611.02779, 2016.

[EAVSG09] Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 289–298, 2009.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 1126–1135. JMLR. org, 2017.

[FMO20a] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In International Conference on Artificial Intelligence and Statistics, pages 1082–1092, 2020.

[FMO20b] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Provably convergent policy gradient methods for model-agnostic meta-reinforcement learning. arXiv preprint arXiv:2002.05135, 2020.
[FXL18] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In Advances in Neural Information Processing Systems, pages 9516–9527, 2018.

[GFL+18] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas L. Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In 6th International Conference on Learning Representations, ICLR, 2018.

[GKS14] Daniel Golovin, Andreas Krause, and Matthew Streeter. Online submodular maximization under a matroid constraint with application to learning assignments. arXiv preprint arXiv:1407.1082, 2014.

[GKW+13] Victor Gabillon, Branislav Kveton, Zheng Wen, Brian Eriksson, and Shanmugavelayutham Muthukrishnan. Adaptive submodular maximization in bandit setting. In Advances in Neural Information Processing Systems, pages 2697–2705, 2013.

[HK15] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis), 5(4):1–19, 2015.

[HKMS19] Hamed Hassani, Amin Karbasi, Aryan Mokhtari, and Zebang Shen. Stochastic continuous greedy++: When upper and lower bounds match. In Advances in Neural Information Processing Systems, pages 13087–13097, 2019.

[HSK17] Hamed Hassani, Mahdi Soltanolkotabi, and Amin Karbasi. Gradient methods for submodular maximization. In Advances in Neural Information Processing Systems, pages 5841–5851, 2017.

[JB11] Stefanie Jegelka and Jeff A Bilmes. Online submodular minimization for combinatorial structures. In ICML, pages 345–352. Citeseer, 2011.

[KB14] Katrin Kirchhoff and Jeff Bilmes. Submodularity for data selection in machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 131–141, 2014.

[KG] Andreas Krause and Daniel Golovin. Submodular function maximization.

[KKT03] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 137–146, 2003.

[KLHK17] Mohammad Karimi, Mario Lucic, Hamed Hassani, and Andreas Krause. Stochastic submodular maximization: The case of coverage functions. In Advances in Neural Information Processing Systems, pages 6853–6863, 2017.

[KMVV15] Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. Fast greedy algorithms in mapreduce and streaming. ACM Transactions on Parallel Computing (TOPC), 2(3):1–22, 2015.

[KSG08] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. Journal of Machine Learning Research, 9(Feb):235–284, 2008.
[LB11] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, pages 510–520. Association for Computational Linguistics, 2011.

[LB12] Hui Lin and Jeff A. Bilmes. Learning mixtures of submodular shells with application to document summarization. In Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, pages 479–490, 2012.

[MBK+15] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrak, and Andreas Krause. Lazier than lazy greedy. In Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

[MHK20] Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Stochastic conditional gradient methods: From convex minimization to submodular maximization. Journal of Machine Learning Research, 21(105):1–49, 2020.

[MKS16] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization. The Journal of Machine Learning Research, 17(1):8330–8373, 2016.

[MZK18] Marko Mitrovic, Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. Data summarization at scale: A two-stage submodular approach. In Proceedings of the 35th International Conference on Machine Learning, pages 3593–3602, 2018.

[MZ15] Vahab Mirrokni and Morteza Zadimoghaddam. Randomized composable coresets for distributed submodular maximization. In Proceedings of the forty-seventh annual ACM symposium on Theory of computing, pages 153–162, 2015.

[RFLK19] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In Advances in Neural Information Processing Systems, pages 113–124, 2019.

[RL17] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In 5th International Conference on Learning Representations, ICLR, 2017.
Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *Advances in Neural Information Processing Systems*, pages 1577–1584, 2009.

Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. ES-MAML: simple Hessian-free meta learning. In *8th International Conference on Learning Representations, ICLR*, 2020.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

Serban Stan, Morteza Zadimoghaddam, Andreas Krause, and Amin Karbasi. Probabilistic submodular maximization in sub-linear time. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3241–3250. JMLR. org, 2017.

Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

UberDataset. Uber pickups in new york city. [https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city](https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city).

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes. Using document summarization techniques for speech data subset selection. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 721–726, 2013.

Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

Yaqing Wang and Quanming Yao. Few-shot learning: A survey. *arXiv preprint arXiv:1904.05046*, 2019.

Yisong Yue and Carlos Guestrin. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems*, pages 2483–2491, 2011.
[YKD+18] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In Advances in Neural Information Processing Systems, pages 7332–7342, 2018.

[ZCHK19] Mingrui Zhang, Lin Chen, Hamed Hassani, and Amin Karbasi. Online continuous submodular maximization: From full-information to bandit feedback. In Advances in Neural Information Processing Systems, pages 9210–9221, 2019.