Submodular Optimization and Data Processing

Kai Wei

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2016

Reading Committee:

Jeff Bilmes, Chair

William Noble

Katrin Kirchhoff

Program Authorized to Offer Degree:
Electrical Engineering

University of Washington

Abstract

Submodular Optimization and Data Processing

Kai Wei

Chair of the Supervisory Committee:
Professor Jeff Bilmes
Department of Electrical Engineering

Data sets are large and and are getting larger. Two common paradigms – data summarization and data partitioning, are often used to handle big data. Data summarization aims at identifying a small sized subset of the data that attains the maximum utility or information, while a core goal of data partitioning is to split the data across multiple compute nodes so that the data block residing on each node becomes manageable. In this dissertation, we investigate how to apply submodularity to these two data processing paradigms.

In the first part of this thesis, we study the connection of submodularity to the data summarization paradigm. First we show that data summarization subsumes a number of applications, including acoustic data subset selection for training speech recognizers [Wei et al., 2014], genomics assay panel selection [Wei et al., 2016], batch active learning [Wei et al., 2015], image summarization [Tschiatschek et al., 2014], document summarization [Lin and Bilmes, 2012], feature subset selection [Liu et al., 2013], etc. Among these tasks, we perform case studies on the former three applications. We show how to apply the appropriate submodular set functions to model the utility for these tasks, and formulate the corresponding data summarization task as a constrained submodular maximization, which admits an efficient greedy heuristic for optimization [Nemhauser et al., 1978]. To better model the utility function for an underlying data summarization task, we also propose a novel "interactive" setting for learning mixtures of submodular functions. For such interactive learning setting,

we propose an algorithmic framework and show that it is effective for both the acoustic data selection and the image summarization tasks. While the simple greedy heuristic already efficiently and near-optimally solves the constrained submodular maximization, data summarization tasks may still be computationally challenging for large-scale scenarios. To this end, we introduce a novel multistage algorithmic framework called MultiGreed, to significantly scale the greedy algorithm to even larger problem instances. We theoretically show that MultGreed performs very closely to the greedy algorithm and also empirically demonstrate the significant speedup of MultGreed over the standard greedy algorithm on a number of real-world data summarization tasks.

In the second part of this thesis, we connect submodularity to data partitioning. We first propose two novel submodular data partitioning problems that we collectively call Submodular Partitioning. To solve submodular partitioning, we propose several novel algorithmic frameworks (including greedy, majorization-minimization, minorization-maximization, and relaxation algorithms) that not only scale to large datasets but that also achieve theoretical approximation guarantees comparable to the state-of-the-art. We show that submodular partitioning subsumes a number of machine learning applications, including load balancing for parallel systems, unsupervised image segmentation, and intelligent data partitioning for parallel training of statistical models. We empirically evaluate the submodular partitioning formulation on the latter two tasks. For both cases, we demonstrate the appropriate choice of submodular utility model and the corresponding submodular partitioning formulation. Empirical evidence suggests that the proposed submodular partitioning framework is effective for both task.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Firstly, I would like to thank my Ph.D. adviser Professor Jeff Bilmes, who guided me into the realm of submodularity and machine learning. I feel extremely lucky to work with Jeff on a number of challenging but exciting research problems. Thanks to Jeff's very broad research interests, I got the chance to work on a variety of research and application areas. I particularly enjoy having research meetings with Jeff, since insightful and useful discussions always come out of such meetings. I really appreciate all of Jeff's supports for my conference travels, which, I believe, are essential experience and very beneficial to my Ph.D. career. With Jeff's supervision, my research is kept on a proper balance between practice and theory. More importantly, I learned from Jeff about how to perform research independently and professionally. Overall, Jeff has been an amazing mentor academically and a very supportive friend personally. I feel blessed to have Jeff as my advisor!

I would also like to thank my thesis committee members: Prof. William Noble, Prof. Katrin Kirchhoff, Prof. Maryam Fazel, and Prof. Thomas Rothvoss. I really appreciate all the wonderful comments and the great efforts on help improving this thesis.

I would also like to thank my collaborators – Rishabh Iyer, Yuzong Liu, Max Libbrecht, Sameer Singh, Chris Bartel, Shengjie Wang, Wenruo Bai, Chandrashekhar Lavania, and Udhay Nallasamy for working together on all projects. It was great experience working with all of you, and I learned a lot from such experience. I would also like to thank all the past and present MELODI lab members – John Halloran, Hui Lin, Rishabh Iyer, Galen Andrew, Bethany Herwaldt, Shengjie Wang, Wenruo Bai, Chandrashekhar Lavania, Tianyi Zhou, Eric Xie, Sebastian Tschiatschek, Rahul Kidambi, Jennifer Gillenwater, Sunil Thulasidasan. Thank you all for the very helpful discussions and for making my Ph.D. life a lot of fun.

I would like to express my gratitude to the sources of my research funding. My research was funded by IARPA with grant No. FA8650-12-2-7263 and later by the NIH under No. R01GM103544.

I am also very grateful for the the tremendous support and encouragement given by my parents Xiangsheng Wei and Guirong Yu. Finally, I would like to dedicate this thesis to Hui Cheng for your endless love and support over the past 8 years. My Ph.D. journey would not have been as joyful and smooth without your accompany.

DEDICATION

to Hui Cheng for her unending support.

Chapter 1

INTRODUCTION

1.1 Background

Recently, data sets are large and are getting even larger. Many of the present-day machine learning systems are trained on vast amounts of data. For example, the state-of-the-art speech recognition systems are often trained on thousands of hours of acoustic data [Schalkwyk et al., 2010]. Larger training data sets often lead to gains in system performance. On one hand, big data is particularly useful as "there is no data like more data." On the other hand, there are well-known problems associated with the ever-increasing data sets: First, larger data sets place greater demands on available computational resources, such as storage and CPU cycles. Second, existing software infrastructure often needs to be adapted to be able to process ever-larger data sets, which requires developer time and experts. Third, the gains in system performance achieved by increasing training data sets are often sublinear: after an initial increase the gains becomes smaller, a phenomenon known as diminishing returns. More importantly, the difficulty of handling big data becomes more aggravated especially nowadays as we are nearing the end of Moore's law [Thompson and Parthasarathy, 2006] and the hardware performance for computing speed has unfortunately not significantly improved since 2003.

One solution to the big data challenge is to carefully choose an informative and representative subset of the data that retains as many of the benefits of the large data set as possible, while simultaneously minimizing resource requirements. We call this problem *data summarization*. A good data summarization strategy should be able to produce a relatively small sized subset that becomes much easier to manage and process, while the summarized subset still attains the most utility or information about the entire data set.

An alternative solution, on the other hand, is to utilize the parallel and distributed computing paradigm. One common approach to achieve parallelism is to split the data into chunks, each of which resides on a compute node. This is the idea behind many parallel machine learning approaches such as ADMM [Boyd et al., 2011] and distributed neural network training [Povey et al., 2014], just to name a few. Such parallel schemes are often performed where the data samples are distributed to their compute nodes in an arbitrary or random fashion. However, there has been apparently been very little work on how to split the data more intelligently. In this case, an interesting goal is to identify a good partitioning of the big data, from which a more efficient or effective parallel learning scheme is achieved. We call this problem *data partitioning*.

In this dissertation, we address both the data summarization and the data partitioning problem based on submodular functions. Submodular functions, often used in economics, operations research, or (more recently) machine learning, are a special class of set functions that satisfy the property of diminishing returns. Defined over a finite ground set V, a set function $f : 2^V \rightarrow \mathbb{R}_+$ is said to be *submodular* [Fujishige, 2005], if for any $a \in V$ and subsets $A \subseteq B \subseteq V$, f satisfies the follows:

$$f(\{a\} \cup A) - f(A) \geq f(\{a\} \cup B) - f(B). \tag{1.1}$$

Certain subclasses of submodular functions can be optimized easily with theoretical performance guarantees. The optimization algorithms, moreover and very importantly, are scalable to very large data sets. A more detailed survey of the submodular function optimization algorithms is given in Section 1.5.

In addition to their amenable optimization properties, submodular functions naturally measure the amount of information that lies within a given set $A \subseteq V$, what is actually meant by "information" depends very much on the particular function and the application. To name a few, given a collection of random variables $X_1, \ldots, X_n$, where $n = V$, the entropy function $f(A) = H(\cup_{a \in A} X_a)$ is submodular. The rank of a subset A of columns of a matrix is also submodular and can be seen as representing information as the dimensionality of the

vector space spanned by the vectors indexed by A. A survey on broad classes of submodular functions that naturally capture the various notions of representativeness, coverage, diversity, and information is given in Section 1.4.

1.2 Methodology

The methodology that we adopt to address both the data summarization and the data partitioning problems is to utilize the appropriate class of submodular functions to first model the utility of a data set, and then formulate the corresponding problem as combinatorial optimization with the submodular utility function as the objective. For the case of data summarization, the combinatorial optimization problem seeks to identify a small sized subset that satisfies certain summarization criterion, while the utility value modeled by the submodular function is maximized. Let V denote the ground set of data, and $f : 2^V \rightarrow \mathbb{R}_+$ denote the submodular function measuring the utility of a data set. The data summarization problem can be naturally formulated as follows:

$$\max_{A \in \mathcal{C}} f(A), \tag{1.2}$$

where $\mathcal{C}$ is the feasible set of subsets restricted by the summarization constraint. A simple and often commonly used choice of $\mathcal{C}$ in applications is defined via the cardinality constraint, i.e., $\mathcal{C} = \{A \subseteq V : |A| \leq k\}$ with some size constraint $k < n$. More sophisticated choices of defining constraints for $\mathcal{C}$ include matroid constraint, knapsack constraint, or even intersection of multiple knapsacks and/or matroids. The feasibility set $\mathcal{C}$ controls the properties of the resulting summary, while the submodular function f judges the quality of the summary. In a word, Problem 1.2 asks for the summary set A^* with the maximum utility value that also abides by the feasibility constraints defined via $\mathcal{C}$. Given the flexibility of the choice of $\mathcal{C}$, a large number of data summarization tasks can be naturally formed in such form of optimization. As we will see, this formulation will be utilized to approach all of the case studies that we consider for data summarization.

Following the similar methodology, we approach the data partitioning scenario by using

submodular functions as the model. Given V as the ground set of the data, let the set of sets $\pi = (A_1^\pi, A_2^\pi, \cdots, A_m^\pi)$ be a partition of a finite set V (i.e, $\cup_i A_i^\pi = V$ and $\forall i \neq j, A_i^\pi \cap A_j^\pi = \emptyset$), and Π refers to the set of all partitions of V into m blocks. Denoting a function $F : \pi \to \mathbb{R}_+$ that measures the goodness of each partitioning choice (defined in detail later). The data partitioning problem may be formulated as the following combinatorial optimization:

$$\max_{\pi \in \Pi} F(\pi). \tag{1.3}$$

Since the quality of a partition often relies on the quality of all its blocks, the definition of $F(\pi)$ may be defined in many ways given the multi-criteria structure of the problem. Let $f : 2^V :\to \mathbb{R}_+$ be the submodular utility function that measures the goodness of each data subset, a simple design of the partition function could be simply the average of the utility scores across its blocks:

$$F_{\text{avg}}(\pi) = \frac{1}{m} \sum_{i=1}^{m} f(A_i^\pi). \tag{1.4}$$

Another criterion would be to consider only the worst-case utility, leading to the formulation:

$$F_{\min}(\pi) = \min_i f(A_i^\pi). \tag{1.5}$$

An even more general objective could be taking the convex combination of the two terms leading to

$$F_\lambda^{\min}(\pi) = \lambda F_{\text{avg}}(\pi) + (1 - \lambda) F_{\min}(\pi), \tag{1.6}$$

where $0 \leq \lambda \leq 1$. In general, all such definitions of the partition function measure an aspect of the quality of the partitioning either via the averaged quality, worst-case quality, or a trade-off of these two. Maximizing F aims at finding a partitioning π such that the overall blocks' quality is optimized. In addition to naturally modeling the utility, information, and diversity of a data set, submodular functions are also often used to capture the notion of cooperative cost [Jegelka and Bilmes, 2011] and computational load [Li et al., 2015]. In such cases, a natural goal is to find a partitioning of the data such that the costs of the resulting blocks are minimized. Mathematically, the optimization has the following form:

$$\min_{\pi \in \Pi} F(\pi), \tag{1.7}$$

where the partition function F could be the averaged-case cost F_{avg} (as defined in Eqn 1.4), the worst-case cost $F_{\max}$ with the following form:

$$F_{\max}(\pi) = \max_i f(A_i^\pi), \tag{1.8}$$

or a combination of the two:

$$F_\lambda^{\max}(\pi) = \lambda F_{\text{avg}}(\pi) + (1-\lambda) F_{\max}(\pi). \tag{1.9}$$

As we will see, both Problem 1.3 and 1.7 in terms of the different choices of the partition function F are useful in machine learning applications such as intelligent data partitioning for distributed statistical learning problems, computational load balancing, and unsupervised image segmentation.

The success of this methodology for both data summarization as well as data partitioning significantly hinges on the accuracy of the modeling and the efficiency of the resulting optimization. As we will see in this dissertation, our aim is to address both components utilizing the tool of submodularity.

1.3 Overview

For data summarization, we perform three concrete case studies on (1) speech data subset selection, (2) genomics assay panel selection, and (3) batch active learning. For data partitioning, we, on the other hand, perform one case study on intelligent data partitioning for distributed machine learning.

1.3.1 Speech Data Subset Selection

The goal of speech data subset selection is to choose a subset of informative and representative acoustic data, on which a high-performance speech recognizer can be trained. The data

selection procedure is often performed on thousands of hours of acoustic data, which consists of over a million speech samples. The focus of this case study is on both the modeling of the utility function for training a speech recognizer and the scalability of the data selection procedure. As we will see in Chapter 2, submodularity naturally address both challenges. We propose several variants of the submodular functions that handle both the medium- and large-scale scenarios of the problem. We also categorize the data selection scenario as supervised versus unsupervised depending whether the transcription of the training data is accessible at the stage of selection. We extensively evaluate on both the phone recognition and the more realistic large vocabulary continuous speech recognition tasks. Significant and consistent improvements over the baseline approaches are achieved by the proposed submodular optimization framework.

1.3.2 Genomics Assay Panel Selection

As for the genomics assay panel selection problem, the goal is to identify a small sized set of genomics assays to perform so that most types of the DNA activities can be captured while the cost of the performed genomics assays is minimized. Given a chosen set of genomics assays, it is also challenging to systematically evaluate the quality of the selected set. In Chapter 3, we concentrate on both the modeling of the utility function for capturing DNA activities and the mechanism for evaluating the quality of any chosen set of genomics assays. We show that using a simple variant of the submodular function as the model would result in a choice of the genomics assays that almost exactly recapitulates the set chosen by biologists using their domain knowledge. We also developed an evaluation framework for the selected set of genomics assays. We focused on three of the most common downstream applications of genomics data sets: (1) imputing assays that haven't been performed, (2) locating functional elements such as promoters and enhancers, and (3) annotating the genome using a semi-automated method. Consistent and significant improvements are achieved by the proposed approach on all three evaluation metrics.

1.3.3 Batch Active Learning

In our third case study, we focus on the batch active learning problem, where, given a collection of unlabeled data samples, the goal is to label a small sized subset of the samples such that the maximum performance gain is achieved by retraining a machine learning classifier with the newly labeled samples. Batch active learning is a classic machine learning problem and has been extensively studied by a number of authors. Several have established the connection of submodularity to this problem [Hoi et al., 2006, Cuong et al., 2010, Guillory and Bilmes, 2010, Chen and Krause, 2013, Golovin and Krause, 2010]. Our focus there is to show the submodularity in the log likelihood function for certain machine learning classifiers, which, thereby, directly connects the submodularity to the utility of training machine learning classifiers and the batch active learning problem. In particular, we show, for simple classifiers such as the Naïve Bayes classifier and the Nearest Neighbor classifier, the log likelihood of the data set for training such classifiers can be naturally formulated as the submodular functions, as a result, leading to the corresponding data summarization as constrained submodular maximization. Furthermore, we apply this framework to active learning and propose a novel scheme called *filtered active submodular selection* (FASS), where we combine the uncertainty sampling method with a submodular data subset selection framework. We extensively evaluate the proposed framework on text categorization and handwritten digit recognition tasks with four different classifiers, including deep neural network (DNN) based classifiers. Empirical results indicate that the proposed framework yields significant improvement over the state-of-the-art algorithms on all classifiers.

1.3.4 Interactive Learning of Mixtures of Submodular Functions

Next, we move a step forward on improving the goodness of the submodular utility model. Instead of hand designing the appropriate submodular utility model using intuition or domain knowledge, we study the problem of automatically learning the submodular utility model in Chapter 5. Given the variety of information that may be captured by different submodular

functions, it is often desirable to design a family of submodular function components and learn the utility set function for an underlying task as a non-negative weighted mixture of these fixed submodular components, where the learned weight represents the importance of information expressed by the corresponding submodular function. While the offline version of this learning problem has been extensively studied in the literature, the focus of our study here is to alleviate the bottleneck of the data collection challenge so that an accurate submodular utility model may be learned in a more feasible manner. We define a novel "interactive" setting for learning the mixtures by adaptively collecting training data, for which we show a number of applications in data summarization, including speech data subset selection, image summarization, and personalized media recommendation. We also provide an integrated framework for solving this problem and empirically demonstrate the efficacy of this framework on several of the aforementioned summarization tasks.

1.3.5 Fast Multi-stage Submodular Maximization

In addition to improving the modeling accuracy of the submodular functions, we address the optimization component of the submodular function based data summarization paradigm. As previously discussed, the submodular framework is often formulated as constrained submodular maximization. In the theory literature for submodular maximization, the simple greedy heuristic is often the optimization technique that one utilizes for solving such combinatorial optimization problems. Besides its simplicity and efficiency for the implementation, the greedy algorithm is often guaranteed to yield a near-optimal solution. On the other hand, many advanced machine learning algorithms that we need to use to process large data sources, in some cases, are still too computationally costly for the amount of data. As we will see in Chapter 6, we address the computational challenge of the greedy algorithm by proposing a multi-stage submodular maximization framework, which at each stage we apply an approximate greedy procedure to maximize surrogate submodular functions. The surrogates serve as proxies for a target submodular function but require less memory and are easy to evaluate. Comparable theoretical guarantees are achieved by the proposed framework

while significant speedup over the standard greedy algorithm is achieved.

1.3.6 Mixed Robust/Average Submodular Partitioning

In the second part of this dissertation, we study the data partitioning paradigm for the big data challenge. Unlike the data summarization where the optimization algorithms for Problem 1.2 have been extensively studied in the literature, the combinatorial optimization formulation used for the data partitioning has drawn less attention in the machine learning community. To this end, we investigate two novel mixed robust/average-case submodular data partitioning problems (Problem 1.3 with $F^{\lambda}_{\min}$ and Problem 1.7 with $F^{\lambda}_{\max}$) that we collectively call *Submodular Partitioning*. These problems generalize purely the robust as well as the average-case instances of the problem. In Chapter 7, we propose to solve the submodular partitioning problems by several new algorithms (including greedy, majorization-minimization, minorization-maximization, and relaxation algorithms) that not only scale to large datasets but that also achieve theoretical approximation guarantees comparable to the state-of-the-art.

1.3.7 Distributed Machine Learning via Intelligent Data Partitioning

Armed with the theoretical advance in the submodular partitioning problems, we perform a case study on the problem of training data partitioning for parallel learning of statistical models. Motivated by the connection of submodularity to the log likelihood function for certain machine learning classifiers, we utilize submodular functions to model the utility of data subsets for training machine learning classifiers and formulate this problem mathematically as submodular partitioning (Problem 1.3). We utilize the simple greedy optimization procedure as proposed in Chapter 7 to near-optimally solve the submodular partitioning problem. We empirically demonstrate the efficacy of the proposed algorithm to obtain data partitioning for distributed optimization of convex and deep neural network objectives. Empirical evidence suggests that the intelligent data partitioning produced by the proposed

framework leads to faster convergence in the case of distributed convex optimization, and better resulting models in the case of parallel neural network training.

1.3.8 Road Map of This Thesis

In the rest of Chapter 1 we introduce basics about submodular functions as well as the existing algorithms for optimizing submodular functions under various forms of constraints.

We structure the remaining chapters into two parts: (1) data summarization, and (2) data partitioning. In Part I, we concentrate on the problem of data summarization. We first give case studies on three concrete applications of data summarization, including speech data subset selection (Chapter 2), genomics assay panel selection (Chapter 3), and batch active learning (Chapter 4). In Chapter 5, we study an interactive learning scheme for improving the accuracy of the submodular utility model for data summarization tasks. We address the scalability of the data summarization algorithms in Chapter 6. In Part II, we investigate the problem of data partitioning. We first study efficient and/or theoretically tight algorithmic frameworks for solving the submodular data partitioning problems in Chapter 7. We next perform a case study of the submodular data partitioning paradigm on data parallelism for distributed statistical machine learning (Chapter 8).

Chapter 2 was originally presented in three conference papers:

- K. Wei, Y. Liu, K. Kirchhoff, J. Bilmes. *Using Document Summarization Techniques for Speech Data Subset Selection*, In NAACL 2013.

- K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes. *Unsupervised Submodular Subset Selection for Speech Data*, In ICASSP 2014.

- K. Wei, Y. Liu, K. Kirchhoff, C. Bartel, and J Bilmes. *Submodular Subset Selection for Large-Scale Speech Training Data*, In ICASSP 2014.

Chapter 3 is being prepared for submission. A pre-print of this work is available online:

- K. Wei, M. Libbrecht, J. Bilmes, and W. Noble. *Choosing panels of genomics assays using submodular optimization*, in bioarxiv 2016.

Chapter 4 was originally published as a conference paper:

- K. Wei, R. Iyer, and J. Bilmes. *Submodularity in Data Subset Selection and Active Learning*, in ICML 2015.

Chapter 5 is being prepared for submission.
Chapter 6 was originally presented in a conference paper:

- K. Wei, R. Iyer, J. Bilmes. *Fast Multi-stage Submodular Maximization*, in ICML 2014.

Chapter 7 was originally presented in a conference paper:

- K. Wei, R. Iyer, S. Wang, W. Bai, J. Bilmes. *Mixed Robust/Average Submodular Partitioning: Fast Algorithms, Guarantees, and Applications*, in NIPS 2015.

Chapter 8 was originally published as a workshop paper:

- K. Wei, R. Iyer, S. Wang, W. Bai, J. Bilmes. *How to Intelligently Distribute Training Data to Multiple Compute Nodes: Distributed Machine Learning via Submodular Partitioning*, in NIPS LearningSys Workshop 2015.

1.4 Submodular Functions: Definition and Examples

Submodularity plays an important role in combinatorial optimization problems. Traditionally studied in mathematics, economics, and operations research, submodularity has recently drawn more attention in the recent advances of many machine learning tasks [Zheng et al., 2014, Hoi et al., 2006, Shamaiah et al., 2010, Prasad et al., 2014, Krause et al., 2008b, Das and Kempe, 2011, Kempe et al., 2003, Gabillon et al., 2013, Chen and Krause, 2013, Reed and Ghahramani, 2013, Tschiatschek et al., 2014, Singla et al., 2014, Shinohara, 2014, Liu et al., 2013, Wei et al., 2014, Lin and Bilmes, 2011, Lin and Bilmes, 2012]. In this section,

we introduce the background on submodular functions, which serves as the foundation for understanding the remainder of the dissertation.

1.4.1 Definitions for set functions

Submodular functions comprise a special class of set functions. Let's first define what a set function is. A *set function* $f : 2^V \to \mathbb{R}$ is a function that maps from any subset $A \subseteq V$ of the underlying ground set V to any real value $f(A)$. Essentially, a set function f assigns a real value for any subset $A \subseteq V$. Intuitively, defining f potentially requires $2^{|V|}$ different valuations for each subset of V. Optimizing a set function f would involve enumerating all possible subsets of V to find the subset with the optimal function value, which has a time complexity of $2^{|V|}$ function valuations. This is clearly not feasible when the ground set size $|V|$ is large. Fortunately, as we will see, the optimization of a set function f becomes much easier as the set function satisfies the notion of "diminishing returns", or equivalently, submodularity. Before formally introducing the definition of submodularity, we first define some useful properties of a set function:

- **normalized**: $f(\emptyset) = 0$
- **monotonically non-decreasing**: $f(A) \leq f(B), \forall A \subset B$.
- **monotonically non-increasing**: $-f$ is monotonically non-decreasing.
- **non-negative:** $f(A) \geq 0, \forall A \subseteq V$.
- **symmetric:** $f(A) = f(V \setminus A), \forall A \subseteq V$.

The set functions considered in this dissertation are assumed to be normalized and non-negative, unless stated otherwise. The special structure of a set function such as monotonicity and symmetry may be exploited to yield more efficient and/or tighter optimization algorithms. When the monotonicity of a set function is assumed, we always assume that the function satisfies the monotonically non-decreasing property, unless stated otherwise.

1.4.2 Equivalent definitions for submodular functions

A submodular function $f : 2^V \rightarrow \mathbb{R}$ is a special class of set functions that satisfy the *diminishing returns* as follows:

$$f(\{a\} \cup A) - f(A) \geq f(\{a\} \cup B) - f(B), \tag{1.10}$$

for any $A \subseteq B \subseteq V, a \in V$. To ease of the notations, we write the marginal gain of an item a on any subset $A \subset V$ as

$$f(a|A) \triangleq f(\{a\} \cup A) - f(A).$$

Eqn 1.10 implies that the incremental gain of any item $a \in V$ always diminishes as the set, on which a is conditioned, grows from A to B. Another equivalent definition of submodularity has also been used in the literature and is given as: a set function f is said to be submodular if it satisfies the following:

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B), \tag{1.11}$$

for any two subsets $A, B \subseteq V$.

In fact, one may come up with many more equivalent definitions of submodularity as shown in the following Proposition.

Proposition 1 ([Nemhauser et al., 1978])**.** *Each of the following statements is equivalent and defines a submodular function.*

- $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$, *for any* $A, B \subseteq V$.
- $f(a|A) \geq f(a|B)$ *for any* $a \in V$ *and* $A \subseteq B \subseteq V$.
- $f(a|A) \geq f(a|A \cup a')$, *for any* $A \subset V$, $a \in V \setminus A$, *and* $a' \in V$.
- $f(C|A) \geq f(C|B)$, *for any* $A \subseteq B \subseteq V$ *and any* $C \subseteq V \setminus B$.
- $f(B) \leq f(A) + \sum_{b \in B \setminus A} f(b|A)$, *for any* $A \subseteq B \subseteq V$.

- $f(A) \leq f(B) + \sum_{a \in A \setminus B} f(a|B) - \sum_{b \in B \setminus A} f(b|A \cup B \setminus b)$, *for any* $A, B \subseteq V$.

These equivalent definitions of submodularity can also be viewed as the properties that a submodular function needs to satisfy. As we will see, these properties are very useful in deriving optimality guarantees in the combinatorial optimization problems involving a submodular function.

1.4.3 Definitions for supermodular and modular functions

Given the definition of a submodular function, we can also define a supermodular function and a modular function. A set function h is said to be *supermodular*, if $-h$ is submodular. In other words, h always satisfies the following:

$$h(a|A) \leq h(a|B), \forall A \subseteq B \subseteq V, a \in V \setminus B. \tag{1.12}$$

In fact, a supermodular function may also be defined via the same set of equivalent definitions as given in Proposition 1 with a reversed inequality.

We say that a set function m is *modular*, if m is both submodular and supermodular. Assuming that m is normalized and modular, m, then, always satisfies the following:

$$m(A) = \sum_{a \in A} m(a), \forall A \subseteq V. \tag{1.13}$$

In contrast to the submodular and supermodular functions, the modular function, by such definition, can be uniquely characterized by a set of $|V|$ real values. Therefore, we typically write a modular function as $m : V \to \mathbb{R}$. One may regard a modular function as a discrete analog of linear function, since its valuation for any set $A \subseteq V$ is simply the linear sum over all its included singleton scores.

For consistent notations, we will, for the rest of the thesis, use f to denote a submodular function, use h for a supermodular function, and m for a modular function, unless stated otherwise.

1.4.4 Definition for curvature of a submodular function

We introduce here another interesting and useful construct called *curvature* [Conforti and Cornuejols, 1984, Iyer et al., 2013a, Vondrák, 2010] – the deviation from modularity – of a monotone submodular function. Given a monotone submodular function f, we define $\kappa_f(A)$ as the curvature of f with respect to a set $A \subseteq V$ as follows:

$$\kappa_f(A) = 1 - \min_{v \in V} \frac{f(v|A \setminus v)}{f(v)}. \tag{1.14}$$

$\kappa_f(A)$ lies in the range of $[0, 1]$ and is monotonically non-decreasing in A. The total curvature as defined in [Conforti and Cornuejols, 1984] is then $\kappa_f(V)$. We notate κ_f as the total curvature of a submodular function f. Intuitively, κ_f measures the distance of f from modularity and $\kappa_f = 0$ if and only if f is modular. On the other hand, we say that a submodular function f is fully curved if $\kappa_f = 1$. As we will see, the construct of curvature is particularly useful for refining approximation bounds for a number of submodular optimization problems.

1.4.5 Important properties of submodular functions

In this section, we focus on the properties of submodular functions. In particular we define a number of important operations that are commonly used in practice while preserving the submodularity. These properties, as we will see, are very useful for proving submodularity of a set function and will be repeatedly used in the upcoming chapters of this thesis.

Proposition 2 (Closed under conic combinations)**.** *Given k submodular functions $\{f_i\}_{i=1}^k$ and k mixture weights $\{w_i\}_{i=1}^k$ with $w_i \geq 0, \forall i$, the weighted mixture $f = \sum_{i=1}^k w_i f_i$ is also submodular.*

Proposition 3 (Complement)**.** *Given a submodular function f, its complement function $\bar{f}(A) \triangleq f(V \setminus A)$ is also submodular.*

Proposition 4 (Min between two)**.** *Given any two submodular functions f_1 and f_2, the minimum of the two $f(A) \triangleq \min\{f_1(A), f_2(A)\}$ is submodular, if $f_1 - f_2$ is either monotonically non-decreasing or non-increasing.*

Proposition 5 (Composition of a modular function with a concave function)**.** *Given a modular function $m : V \to \mathbb{R}$ and a continuous concave function $g : \mathbb{R} \to \mathbb{R}$, the composition formed as $f(A) \triangleq g(m(A))$ is submodular.*

Proposition 6 (Restricted to a set)**.** *Given a submodular function f and a set $B \subseteq V$, the restricted set function $f'(A) \triangleq f(A \cap B)$ is submodular.*

Proposition 7 (Composition of a submodular function with a concave function)**.** *Given a monotone submodular function f and monotone concave function g, the composition formed as $f'(A) \triangleq g(f(A))$ is also monotone submodular.*

Corollary 1 (Truncated submodular function)**.** *Given a monotone submodular function f and a constant c, the truncated function $f'(A) \triangleq \min\{f(A), c\}$ is monotone submodular.*

Despite that the above operations nicely preserve submodularity, we would also like to point out some other commonly used operations that often break submodularity below.

Proposition 8 (Operations that generally do not preserve submodularity)**.** *Given two submodular functions f_1 and f_2,* none *of the following can be guaranteed as submodular:*

- $f(A) \triangleq f_1(A) f_2(A)$
- $f(A) \triangleq \frac{f_1(A)}{f_2(A)}$
- $f(A) \triangleq f_1(A) - f_2(A)$
- $f(A) = \max\{f_1(A), f_2(A)\}$
- $f(A) = \max\{f_1(A), f_2(A)\}$

1.4.6 Examples of submodular functions

Having given the properties for proving submodularity, we will show, in this section, a number of submodular functions that one may frequently come across in practice. First we consider the class of submodular functions which are defined via modular functions. Many instances of submodular functions are built upon modular functions despite their simplicity. Some are formed by simply composing a concave function with a modular function, while other forms include more non-linear operations over the modular functions, such as taking the maximum. We summarize the various instances of the submodular functions in such form in the following proposition:

Proposition 9 (Composition of modular functions)**.** *Given any two modular functions $m, m' : V \to \mathbb{R}$, the following set functions are submodular:*

- $f(A) \triangleq -m(A)m'(A)$.
- $f(A) \triangleq \max_{a \in A} m(a)$.
- $f(A) \triangleq \log(1 + m(A))$.
- $f(A) \triangleq 1 - \prod_{a \in A} m(a)$ *with* $0 \leq m(a) \leq 1, \forall a \in V$.
- $f(A) \triangleq 1 - \alpha^{-m(A)}$ *with* $\alpha > 1$.

Next, we introduce another class of submodular functions that are defined via a bipartite graph. Bipartite graphs are those graphs whose vertices can be categorized into two disjoint sets, say V and U, such that each edge connects a vertex $v \in V$ to a vertex $u \in U$.

Formally, a weighted complete bipartite graph $G(V, U, E)$ is defined via three components: (1) V: a set of items, (2) U: a set of objects, and (3) E: the set of all edges connecting between an item $v \in V$ and an object $u \in U$. Let $e_{v,u}$ denote the edge between v and u, and $w_{v,u}$ be the weight associated with this edge. An example of a bipartite graph is illustrated

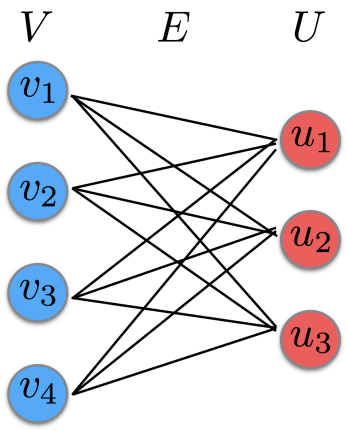


Figure 1.1: An example of a bipartite graph $G(V, U, E)$, where $V = \{v_1, v_2, v_3, v_4\}$ and $U = \{u_1, u_2, u_3\}$.

in Figure 1.1. Define a modular function $m_u(A) \triangleq \sum_{a \in A} w_{a,u}$ that computes the relevance of the set $A \subseteq V$ to the object $u \in U$.

Many variants of submodular functions can be defined via the weighted bipartite graph G. For instance, the well-known set cover function is defined on a special instance of the bipartite graph G where the weights $w_{v,u}$ take only either 0 or 1 indicating the presence of an edge. The set cover function has the following form:

$$f_{\text{sc}}(A) = \sum_{u \in U} \min\{m_u(A), 1\}. \tag{1.15}$$

Here, the modular function $m_u(A)$ is interpreted as the number of edges connecting A and u. The submodularity of f_{sc} is evident as each of its constituent terms is submodular according to Corollary 1.

Defining for each object $u \in U$ the importance level $q_u \geq 0$ and the truncation level c, the generalized set cover function has the following form:

$$f_{\text{g-sc}}(A) = \sum_{u \in U} q_u \min\{m_u(A), c\}. \tag{1.16}$$

We may further generalize this function assuming a set of $|U|$ given concave functions $g_u : \mathbb{R} \to \mathbb{R}$ for each $u \in U$ as follows:

$$f_{\text{fea}}(A) = \sum_{u \in U} q_u g_u(m_u(A)). \tag{1.17}$$

In this dissertation, we call this function *feature based function*, since the set of objects U, on which the function is defined, is often viewed as the set of features that one may wish to cover in the tasks examined in this thesis. Here we define the concave functions in the most general form: each feature is associated with a different concave function, however, it often suffices to use a single concave function in practice. The notion of the features varies as the application considered at hand. $f_{\text{g-sc}}$ is a special instance of f_{fea} with each concave function g_u identical as $g(x) = \min\{x, c\}$. As we will see, the feature based function f_{fea} naturally occurs as the utility model of data set for a variety of summarization tasks. Furthermore, nice interpretations can be given for certain instances of the feature based function in these tasks.

Another class of submodular function also commonly used is called the maximum coverage function, which is also known as the facility location function [Mirchandani and Francis, 1990]. This function has the following form:

$$f_{\text{fac}}(A) = \sum_{u \in U} \max_{a \in A} w_{a,u}. \tag{1.18}$$

The facility location function as defined in Eqn 1.18 implicitly assumes that the importance level q_u for each object $u \in U$ is uniform, and can be easily extended to a weighted mixture specified by a set of non-uniform weights q_u. Since the uniform weighted version as in Eqn 1.18 is often the variant that one uses in practice, we only give the definition in this form here. If $w_{a,u}$ measures the relevance of the item a to the object u, the facility location function f_{fac} valuates for any set $A \subseteq V$ the sum over each object $u \in U$ the maximum relevance scores in A. Intuitively, maximizing this function f_{fac} naturally encourages finding a set A with the diverse coverage over the set of objects U. f_{fac} is submodular since each constituent term $\max_{a \in A} w_{a,u}$ is submodular thanks to Proposition 9.

Another interesting class of submodular function defined over a bipartite graph is called *probabilistic cover function* [Yue and Guestrin, 2011, El-Arini et al., 2009]. Here, the weights over the edges are given a probabilistic interpretation with the assumption that $w_{v,u} \in [0,1], \forall v \in V, u \in U$. The function has the following form:

$$f_{\text{prob}}(A) = \sum_{u \in U} q_u (1 - \prod_{a \in A} (1 - w_{a,u})). \tag{1.19}$$

Let $w_{v,u}$ model the probability that the object u is covered by the item v. Assuming the independence among the items for covering each object u, the probability that an object u is covered by at least one item in A can be written as $1 - \prod_{a \in A}(1 - w_{a,u})$. Hence, $f_{\text{prob}}(A)$ evaluates the weighted expectation of the number of objects in U that are covered by at least an item in A. The submodularity here follows because of Proposition 9.

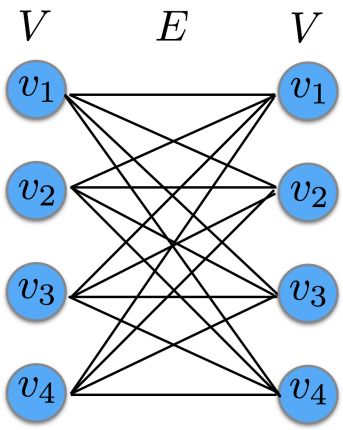


Figure 1.2: An example of a bipartite graph $G(V, U, E)$ for representing a complete similarity graph, where $V = \{v_1, v_2, v_3, v_4\}$.

Lastly, we focus on a different graph – complete similarity graph – and introduce several classes of submodular functions that are defined on such graphs. The complete graph $G_{\text{complete}}(V, E)$ is defined over a set of items V. For any pair of items $v_1 \in V$ and $v_2 \in V$, the graph has an edge $e_{v_1,v_2} \in E$ connecting between them. The weight w_{v_1,v_2} associated with the

edge defines the similarity between the two items. The similarity measure is symmetric, i.e., $w_{v_1,v_2} = w_{v_2,v_1}$ for any pair $v_1, v_2 \in V$. One may also represent the complete similarity graph using a bipartite graph as illustrated in Figure 1.2. The submodular functions previously defined over the bipartite graph easily generalize to the complete similarity graph. As we will see, the variant of the facility location function that is defined on the complete similarity graph, i.e.,

$$f_{\text{fac}}(A) = \sum_{v \in V} \max_{a \in A} w_{v,a}, \tag{1.20}$$

is more commonly used in the applications considered in this paper.

Another class of submodular function defined over the similarity graph is called the *saturated coverage function* [Lin and Bilmes, 2011], which has the following form:

$$f_{\text{sat}}(A) = \sum_{v \in V} \min\{m_v(A), c\}, \tag{1.21}$$

where $m_v(A) = \sum_{a \in A} w_{v,a}$ measures the relevance of the set A to the item $v \in V$, and c is a saturation hyperparameter that controls the level of coverage for each item $v \in V$ by the set A. We would also like to point out that the saturated coverage function, which has been successfully applied in the document summarization task, is a special instance of the generalized set cover function $f_{\text{g-sc}}$.

An interesting class of function that naturally captures the notion of redundancy of a data set can also be defined via a similarity graph as follows:

$$f_{\text{red}}(A) = \sum_{a \in A} \sum_{a' \in A} w_{a,a'}. \tag{1.22}$$

We call this function *redundancy function*. This function is supermodular, which can be easily verified by the definition of supermodularity. Hence $-f_{\text{red}}$ is submodular. Intuitively, a set A of similar items tends to yield a high valuation in $f_{\text{red}}(A)$. In practice, one can encourage the diversity in a data set by maximizing $-f_{\text{red}}$.

We conclude this section by the classic graph cut function as defined below:

$$f_{\text{gc}}(A) = \sum_{a \in A} \sum_{v \in V \setminus A} w_{v,a}. \tag{1.23}$$

The submodularity of f_{gc} can be verified by simply rewriting function as $f_{\text{gc}}(A) = \sum_{v \in V} m_v(A) - f_{\text{red}}(A)$, where the first term is a modular function and the second term f_{red} is supermodular.

1.5 Submodular Function Optimization and Existing Algorithms

Assuming submodularity in a discrete objective, the resulting combinatorial optimization becomes submodular function optimization. In this section, we survey the existing optimization algorithms for various forms of submodular optimization problems. We first define two notions of continuous extension of submodular functions: (1) Lovász extension, and (2) multi-linear extension. The Lovász extension, in nature, is a convex extension of a submodular function and is particularly useful for submodular minimization problems, while the multi-linear extension serves, to some extent, as a concave extension of a submodular function and is often applied in submodular maximization scenarios. We then introduce the semigradients (subgradient and supergradient) of a submodular function. We will see that a submodular function admits efficient computation of a subgradient and a supergradient. They are particularly useful for defining the modular lower and upper bounds of the submodular function. We lastly introduce the notion of a matroid, which is a combinatorial structure that is well connected to submodular functions and is often used for defining the set of feasibility sets in the constrained combinatorial optimization problems. Armed with the continuous extensions of submodular functions and the tool of matroids, we survey the existing algorithms for the various forms of submodular optimization problems, including both the maximization and the minimization scenarios, both the constrained and unconstrained variants, and both the monotone and the non-monotone cases.

1.5.1 Continuous extensions of submodular functions

Lovász extension

A very nice property of a submodular function is that its convex extension can be efficiently defined and is known as *Lovász extension* [Lovász, 1983]. It reveals an interesting connection

of submodular set functions to the continuous convex functions. Given any set function $f : 2^V \to \mathbb{R}$, we give the definition of its Lovász extension $\tilde{f} : [0,1]^n \to \mathbb{R}$ ($|V| = n$) as follows:

Definition 1. *Given any $x \in [0,1]^n$, define a permutation σ_x by ordering its elements in non-increasing order, and thereby a chain of sets $S_0^{\sigma_x} \subset, \ldots, \subset S_n^{\sigma_x}$ with $S_j^{\sigma_x} = \{\sigma_x(1), \ldots, \sigma_x(j)\}$ for $j = 1, \ldots, n$. The Lovász extension $\tilde{f}$ for f is the weighted sum of the ordered entries of x:*

$$\tilde{f}(x) = \sum_{j=1}^{n} x(\sigma_x(j))(f(S_j^{\sigma_x}) - f(S_{j-1}^{\sigma_x})). \tag{1.24}$$

An interesting relationship between submodularity and convexity can be built via the Lovász extension as shown below:

Theorem 1 ([Lovász, 1983])**.** *For any set function $f : 2^V \to \mathbb{R}$ and its Lovász extension $\tilde{f} : [0,1]^n \to \mathbb{R}$, f is submodular if and only if $\tilde{f}$ is convex.*

It is easy to see that the Lovász extension $\tilde{f}$ has the same function valuation as f on vertices of the unit hypercube. Denoting 1_A as the characteristic vector of the set A, it always holds that $\tilde{f}(1_A) = f(A), \forall A \subseteq V$. Given a submodular function f, $\tilde{f}$ is, therefore, a convex extension of f thanks to Theorem 1. The Lovász extension provides a general algorithmic framework for minimizing submodular functions. Given an instance of submodular minimization problem, one can relax the discrete submodular objective that is defined on only the vertices of the unit hypercube to a continuous convex function that is defined on the whole unit hypercube. In the unconstrained or constrained cases where the constraints may be characterized as a convex set, then the relaxed optimization problem becomes a convex optimization, for which there exists efficient and polynomial time solutions [Boyd and Vandenberghe, 2004]. The remaining questions are to devise a rounding technique that rounds the fractional solution output from the relaxation formulation to a vertex on the hypercube. In many cases, efficient rounding techniques can be designed, by which bounded performance loss is introduced, hence, leading to a near-optimal solution of this relaxation scheme.

Multi-linear extension

For the purpose of maximizing a submodular function, a concave extension of a submodular function is needed leading to relaxation algorithms for submodular maximization problems. Unfortunately, the concave closure of submodular functions is NP-hard to evaluate [Vondrák, 2007]. An alternative variant of continuous extension is known as the *multi-linear extension*. Although it is not concave, the multi-linear extension has some concave-like properties leading to efficient maximization algorithms for the resulting relaxation formulations. First introduced by [Vondrák, 2008], the multi-linear extension $\hat{f} : [0,1]^n \to \mathbb{R}$ has the following definition.

Definition 2. *Given a set function $f : 2^V \to \mathbb{R}$, its multi-linear extension $\hat{f} : [0,1]^n \to \mathbb{R}$ has the form:*

$$\hat{f}(x) = \sum_{A \subseteq V} f(A) \prod_{a \in A} x_a \prod_{a \in V \setminus A} (1 - x_a). \tag{1.25}$$

The multi-linear extension $\hat{f}$ has a nice probabilistic interpretation: $\hat{f}(x)$ evaluates the expected set function valuation $f(A_x)$ with A_x being the random set that each item $a \in A_x$ is sampled with probability x_a. Denoting 1_A the characteristic vector for the set $A \subseteq V$, the multi-linear extension $\hat{f}$ agrees with the set function f on the vertices of the hyper-cube, i.e., $\hat{f}(1_A) = f(A)$. Although evaluating $\hat{f}(x)$ for general x requires an exponential number of queries on the function f, concentrated estimate of the multi-linear extension $\hat{f}$ can be achieved using a polynomial number of evaluation of the function f on random subsets according to the Chernoff bounds.

Moreover, useful conditions on the second-order derivative of a multi-linear function $\hat{f}$ can be shown if its set function counterpart f satisfies submodularity.

Theorem 2. *[Calinescu et al., 2007] Given a set function $f : 2^V \to \mathbb{R}$ and its multi-linear extension $\hat{f} : [0,1]^n \to \mathbb{R}$. f is submodular if and only if $\hat{f}$ satisfies the following:*

$$\frac{\partial^2 \hat{f}(x)}{\partial x_i \partial x_j} \leq 0, \forall i, j, x \in [0,1]^n. \tag{1.26}$$

Such a design of the continuous extension for a submodular function has proved to be particularly useful for maximizing a submodular function under a matroid constraint. The best and tightest algorithm for this optimization problem is achieved via a continuous greedy procedure on optimizing the multi-linear extension [Vondrák, 2008].

1.5.2 Discrete semigradients of a submodular function

Convex functions naturally have subgradients (linear lower bounds), while concave functions have supergradients (linear upper bounds). Interestingly, submodular functions have both sub and super gradients. Subgradients of submodular functions are mostly studied by [Fujishige, 2005], while the investigations on supergradients of a submodular function are recently given by [Iyer, 2015, Iyer and Bilmes, 2015].

The subdifferential $\partial_f(Y)$ of the submodular function f at a set $Y \subseteq V$ is defined analogously to the subdifferential of a continuous convex function and has the following form [Fujishige, 2005]:

$$\partial_f(Y) = \{y \in \mathbb{R}^n : f(X) - y(X) \geq f(Y) - y(Y) \text{ for all } X \subseteq V\}. \tag{1.27}$$

Denote a subgradient at Y by $h_Y \in \partial_f(Y)$. The extreme points of $\partial_f(Y)$ may be computed via a greedy algorithm: Let σ be a permutation of V that assigns the elements in Y to the first $|Y|$ positions ($\sigma(i) \in Y$ if and only if $i \leq |Y|$).

Each such permutation defines a chain with elements $S_0^\sigma = \emptyset$, $S_i^\sigma = \{\sigma(1), \sigma(2), \ldots, \sigma(i)\}$, and $S_{|Y|}^\sigma = Y$. An extreme point h_Y^σ of $\partial_f(Y)$ has each entry as

$$h_Y^\sigma(\sigma(i)) = f(S_i^\sigma) - f(S_{i-1}^\sigma). \tag{1.28}$$

$h_Y^\sigma(\sigma(i)) = f(S_i^\sigma) - f(S_{i-1}^\sigma)$. Defined as above, h_Y^σ forms a modular lower bound of f, tight at Y — i.e., $h_Y^\sigma(X) = \sum_{j \in X} h_Y^\sigma(j) \leq f(X), \forall X \subseteq V$ and $h_Y^\sigma(Y) = f(Y)$.

Sharing the similar definition as the subdifferentials of a submodular function f, the superdifferentials $\partial^f(Y)$ of f at Y has the following form:

$$\partial^f(Y) = \{y \in \mathbb{R}^n : f(X) - y(X) \leq f(Y) - y(Y); \text{for all } X \subseteq V\}. \tag{1.29}$$

Two specific forms of supergradients are often used to define the modular upper bounds of a submodular function (we use $m^f_{Y,1}, m^f_{Y,2} \in \partial^f(Y)$):

$$m^f_{Y,1}(X) \triangleq f(Y) - \sum_{j \in Y \setminus X} f(j|Y \setminus j) + \sum_{j \in X \setminus Y} f(j|\emptyset),$$
$$m^f_{Y,2}(X) \triangleq f(Y) - \sum_{j \in Y \setminus X} f(j|V \setminus j) + \sum_{j \in X \setminus Y} f(j|Y). \tag{1.30}$$

Then $m^f_{Y,1}(X) \geq f(X)$ and $m^f_{Y,2}(X) \geq f(X), \forall X \subseteq V$ and $m^f_{Y,1}(Y) = m^f_{Y,2}(Y) = f(Y)$. Hence both $m^f_{Y,1}$ and $m^f_{Y,2}$ are modular upper bounds of f, tight at Y.

[Iyer, 2015] show that the semigradients of a submodular function are very useful to provide scalable algorithms for a number of combinatorial optimization problems involving submodular functions. As we will see in Chapter 7, efficient algorithms with bounded performance guarantees can be designed using the same paradigm for the submodular partitioning problems.

1.5.3 Matroid

In this section, we introduce the notion of a matroid – a combinatorial structure that is well connected to submodular functions. We first give the definition of a matroid. We then give several classes of matroids that are commonly used in practice. We last explain how matroids may be connected to the paradigm of greedy algorithm and submodular functions.

Definition

A matroid is defined via two components: (1) a finite ground set V, and (2) an independence system $\mathcal{I}$ which consists of a collection of independent subsets of V. We denote a matroid as $\mathcal{M}(V, \mathcal{I})$. The independence system $\mathcal{I}$ needs to satisfy the following three conditions so that $\mathcal{M}$ is called a matroid.

1. $\emptyset \in \mathcal{I}$,

2. Given $A \in \mathcal{I}$, $B \in \mathcal{I}, \forall B \subset A$.

3. $\forall A, B \in \mathcal{I}$ with $|A| > |B|$, there exists $a \in A \setminus B$ such that $B \cup \{a\} \in \mathcal{I}$.

The first condition requires that the empty set is included in the set system $\mathcal{I}$. The second condition makes sure that $\mathcal{I}$ is down-closed, i.e., if a set is independent, all its subsets should also be independent. The third condition is known as the exchange property, namely, for any pair of subsets A and B that are not of the same size (assuming $|A| > |B|$), at least one item from the $A \setminus B$ can be added to B still rendering it independent.

For any subset $A \subseteq V$, a *base* of A is the set B_A that is inclusionwise maximally independent subset of A, namely, $B_A \in \mathcal{I}$ and there does not exist any $Z \in \mathcal{I}$ with $B_A \subset Z \subseteq A$. A base of the matroid $\mathcal{M}$ is simply the base of the ground set V.

Proposition 10. *Given a matroid $\mathcal{M}(V, \mathcal{I})$, for any $A \subseteq V$, any two bases of A have the same size.*

This Proposition follows from the third condition of the matroid. It also implies that any bases of the matroid $\mathcal{M}$ should be of the same size.

Another useful concept associated with a matroid is called *rank*. Given a matroid $\mathcal{M}$, the rank of any subset $A \subseteq V$ is the common size of all the bases of A. The *rank function* $r_{\mathcal{M}} : 2^V \to \mathbb{Z}_+$ of a matroid $\mathcal{M}$ is defined as a set function that returns the rank of any subset $A \subseteq V$. More formally, the rank function has the following formulation:

$$r_{\mathcal{M}}(A) = \max\{|X| : X \subseteq A, X \in \mathcal{I}\} = \max_{X \in \mathcal{I}} |A \cap X|. \tag{1.31}$$

For any set $A \subseteq V$, it always holds that $r(A) \leq |A|$, and the equality is met when the set A is independent, i.e., $A \in \mathcal{I}$. Interestingly, the matroid rank function $r_{\mathcal{M}}$ is a submodular set function providing a connection between matroids and submodularity.

Matroids, on the other hand, are often strongly tied with the greedy algorithm, a commonly used heuristic for combinatorial optimization problems. The idea of the greedy algorithm is to iteratively update the solution such that the objective is improved the best. A simple optimization problem involving a matroid is to maximize a modular function under

a matroid constraint. Mathematically, the problem has the following form:

$$\max_{A \in \mathcal{I}} m(A), \tag{1.32}$$

where $m : V \to \mathbb{R}_+$ is a modular function.

Algorithm 1: Greedy algorithm for modular maximization under a matroid constraint.

Input: A modular function $m : V \to \mathbb{R}_+$ and a matroid $\mathcal{M}(V, \mathcal{I})$.
Initialize: $A \leftarrow \emptyset$.
while $\exists a \in V \setminus A : A \cup \{a\} \in \mathcal{I}$ **do**
    $a \in \operatorname{argmax}_{a \in V \setminus A : A \cup \{a\} \in \mathcal{I}} w(a)$.
    $A \leftarrow A \cup \{a\}$.
Output: A.

Given the greedy heuristic as defined in Algorithm 1, a bridge connecting a matroid to the greedy algorithm can be established via the following Theorem:

Theorem 3. *[Edmonds, 1970] Let $\mathcal{I}$ be an independence system. The pair $(V, \mathcal{I})$ is a matroid if and only if for every modular function $m \in \mathbb{R}_+$, Algorithm 1 solves Eqn 1.32 exactly.*

Theorem 3 implies that the greedy heuristic is a defining characteristic of a matroid. We point out that the optimal solution may not be attained by the greedy heuristic for Eqn 1.32 when the modular function m is generalized to be submodular. However, the very same greedy heuristic, as we will see in Section 1.5.5, is still guaranteed to yield a near-optimal solution in the case of a monotone submodular objective.

Matroid examples

In this section, we introduce several examples of matroids that are commonly used in practice. The first example is called *k-uniform matroid*. Given a ground set V, the independence

system for the k-uniform matroid is the set of all subsets that are of size at most k. Mathematically, the k-uniform matroid has an independent system

$$\mathcal{I} = \{A \subseteq V : |A| \leq k\}.$$

The rank function, in this case, has the form:

$$r(A) = \min\{|A|, k\}.$$

Its submodularity is easily verified as it is in the form of concave over modular with the concave function $g(x) = \min\{x, k\}$. Any subset A in a k-uniform matroid is independent if the size $|A|$ is no more than k. This matroid is particularly useful for modeling the cardinality constraint of size k in the optimization problems.

Another class of matroids is called *linear matroid*. Defined on a matrix H of size $m \times n$, the linear matroid $\mathcal{M}(V, \mathcal{I})$ has the ground set V as the set of columns in H. The independence system $\mathcal{I}$ consists of subsets of V such that if $A \in \mathcal{I}$, the column vectors in H indexed by A are linearly independent. The rank function $r(A)$ simply returns the rank of the vector space spanned by the column vectors indexed by A.

Another example of useful matroids is the *partition matroid*. Given a disjoint partitioning of a ground set $V = \{V_1, \ldots, V_m\}$ into m blocks, the independence system of the partition matroid has the following definition:

$$\mathcal{I} = \{A \subseteq V : |A \cap V_i| \leq k_i, \forall i = 1, \ldots, m\}, \tag{1.33}$$

where $k_1, \ldots, k_m \geq 0$ are fixed parameters. The rank function, in this case, can be written as

$$r(A) = \sum_{i=1}^{m} \min\{|A \cap V_i|, k_i\}.$$

The partition matroid generalizes the k-uniform matroid by setting $m = 1$ and $k_1 = k$. Given a disjoint partition of V into m blocks and the fixed parameters $k_1, \ldots, k_m$, the partition matroid asks for an independent set to intersect each block V_i with no more than k_i items.

1.5.4 Submodular Minimization

Armed with the preliminary definitions related to submodular functions, we are ready to survey the existing algorithms for submodular optimization problems. Since the methodologies for minimizing a submodular function are often very different from the paradigms for the maximization case, we survey the optimization algorithms separately for these two cases. In this section, we focus on the submodular minimization problems. We further categorize the problem into unconstrained and constrained cases.

Unconstrained case

The problem is only non-trivial if the submodular function is non-monotone. Analogous to the convex function, any submodular function can be minimized in polynomial time.

The first polynomial time algorithm is given in [Grötschel et al., 1981]. This algorithm employs the ellipsoid method to minimize the convex relaxation of the submodular function via the Lovász extension. Despite its polynomial time complexity, the ellipsoid algorithm is, in general, not practical in practice.

Combinatorial strongly polynomial algorithm was first developed for solving the membership problem for matroid polyhedra (a special case of submodular function minimization) in [Cunningham, 1984]. Built on this framework, combinatorial strongly polynomial algorithms were given for solving the general submodular minimization [Iwata et al., 2001,Schrijver, 2000]. Note that strongly polynomial algorithm given by [Iwata et al., 2001] runs in $O(\gamma n^7 \log n)$ while the complexity of Schrijver's algorithm is $O(n^8 + \gamma n^7)$ with γ being the complexity of function evaluation oracle. The currently fastest known strongly polynomial time algorithm for submodular minimization is given in [Orlin, 2009], which runs with a time complexity of $O(n^5\gamma + n^6)$. In general, these algorithms, although admitting a polynomial running time, are not practical for scaling to applications of interests.

When the submodular function is symmetric, a significantly faster algorithm was proposed by [Queyranne, 1998] for solving the minimization problem with $O(n^3)$ function val-

uations. Another popular algorithm for general submodular minimization is called the minimum-norm-point algorithm [Fujishige et al., 2006], which is always among the best empirical performance. Recently, the first pseudo-polynomial time guarantee is proved for the minimum-norm-point algorithm, which is guaranteed to return the minimizer of f in $O((n^5\gamma + n^7)F^2)$ time with $F = \max_{a \in V}\{|f(a)|, |f(V) - f(V \setminus a)|\}$ [Chakrabarty et al., 2014].

Constrained case

In contrast to the unconstrained submodular minimization problem, the constrained case is much harder. For even the simplest constraints such as size constraint, the problem cannot be solved globally in polynomial time. Most existing work on constrained submodular minimization assumes that the submodular objective is monotone.

In the case of cardinality lower bound constraint, [Svitkina and Fleischer, 2008] show that the problem is information theoretically hard to approximate within $o(\sqrt{\frac{n}{\log n}})$ and give a $(5\sqrt{\frac{n}{\log n}}, 1/2)$ bicriteria sampling-based approximation algorithm. In the case of equal cardinality constraint, [Nagano et al., 2011] give a minimum-norm based algorithm, which exactly solves the problem for certain choices of cardinality constraints. Recently, [Iyer et al., 2013b] propose a majorization-minimization framework that handles the constrained submodular minimization problem for general choices of constraints. Their idea is to iteratively derive a tight modular upper bound of the submodular function and to transform the problem to its modular function counterpart, which can always be solved exactly. Bounded performance guarantees in terms of the curvature of the submodular function are given by this paradigm for many constrained submodular minimization problems.

Another general paradigm for solving the constrained submodular minimization problems is to relax the submodular function as a convex function via its Lovász extension [Iyer et al., 2014], where they propose simple rounding techniques that yield bounded rounding loss for general constaints such as the matroid constraint, set covers, paths, cuts, and matchings. Other variants of constrained submodular minimization problems can be found in these

work [Goel et al., 2009, Iwata and Nagano, 2009].

1.5.5 Submodular Maximization

Unlike the minimization case, submodular maximization subsumes a number of NP-hard problems, e.g., max-k-cover and graph cut, hence is generally NP-hard. Fortunately, there often exists low order polynomial algorithms for most cases of submodular maximization for both the unconstrained and the constrained scenarios. Contrary to the minimization case, we give more detailed survey on the algorithms for submodular maximization, since some of the algorithms are actually implemented in our work for several data summarization applications. Moreover, some of the algorithmic ideas are also leveraged to devise even more efficient algorithms for submodular data summarization and submodular data partitioning problems.

Problem 1 (Unconstrained Submodular Maximization)**.**

$$\max_{A \subseteq V} f(A) \tag{1.34}$$

In the unconstrained case, the problem is only non-trivial if the submodular function is non-monotone. [Feige et al., 2011] show that the problem in the general non-monotone case is information theoretically hard to approximate better than 1/2. A randomized linear time bi-directional greedy procedure (see Alg. 2) is given in [Buchbinder et al., 2012]. This algorithm performs only one pass over the entire data V. In each iteration i, the algorithm maintains two candidate solutions X_{i-1} and Y_{i-1} simultaneously, examines the gain of either adding v_i to X_{i-1} or the gain by removing v_i from Y_{i-1}, and performs either operation with a randomized strategy where the probability of choosing either operation is proportional to its benefit for improving the objective value. Albeit its simplicity and the efficiency, Alg. 2 is guaranteed to output a solution X_n that attains a 1/2-approximation factor on expectation to Problem 1. Therefore, this algorithm is the first optimal algorithm for the unconstrained submodular maximization.

Algorithm 2: Bi-directional greedy algorithm for Problem 1 [Buchbinder et al., 2012]

Input: An ordered ground set $V = \{v_1, \ldots, v_n\}$.
Initialize: $X_0 \leftarrow \emptyset$, $Y_0 \leftarrow V$.
for $i = 1, \ldots, n$ **do**
  $a_i \leftarrow \max\{0, f(v_i|X_{i-1})\}$.
  $b_i \leftarrow \max\{0, -f(v_i|Y_{i-1} \setminus v_i)\}$.
  Sample a number ρ uniformly and independently at random from $[0, 1]$.
  if $\rho \leq \frac{a_i}{a_i+b_i}$ **then**
    $X_i \leftarrow X_{i-1} \cup \{v_i\}$.
    $Y_i \leftarrow Y_{i-1}$.
  $X_i \leftarrow X_{i-1}$.
  $Y_i \leftarrow Y_{i-1} \setminus \{v_i\}$.
Return: X_n (or equivalently Y_n).

Recently, there has been some advances in further scaling up the bi-directional greedy algorithm [Pan et al., 2014] as well as designing optimal deterministic algorithms [Buchbinder and Feldman, 2015]. Several parallel variants of Alg. 2 are proposed in [Pan et al., 2014] to address the scalability issue, where the data set size is simply too large to fit into a single compute node. The parallel schemes either scale up the algorithm at the cost of a weaker approximation guarantee or the cost of synchronization. On the theoretical side, a deterministic algorithm has been shown in [Buchbinder and Feldman, 2015] to achieve the optimal 1/2-approximation for Problem 1.

Next we study the constrained case of submodular maximization. Since monotone submodular functions are mostly used in the applications examined in this paper, we will concentrate on the monotone case and defer the discussion on the non-monotone case to the end of this section. A simple but extremely useful constraint is the cardinality constraint leading to Problem 2.

Problem 2 (Cardinality Constrained Submodular Maximization)**.**

$$\max_{|A|\leq k, A\subseteq V} f(A) \tag{1.35}$$

Algorithm 3: Greedy algorithm for Problem 2 [Nemhauser et al., 1978]

1 **Input:** k, V, and f.
2 **Initialization:** $A \leftarrow \emptyset$.
3 **while** $|A| < k$ **do**
4 $\quad a^* \leftarrow \text{argmax}_{a\in V\setminus A} f(a|A)$.
5 $\quad A \leftarrow A \cup a^*$.
6 **Output:** $\hat{A} \leftarrow A$.

A simple greedy heuristic (see Alg. 3) is first proposed in [Nemhauser et al., 1978] to near-optimally solve the problem. The greedy algorithm, in each iteration i, finds the item a^* with the maximum marginal gain conditioned on current solution A and adds it to the solution as $A \leftarrow A \cup a^*$. Despite its simplicity, the algorithm attains the following approximation guarantee.

Theorem 4. *[Nemhauser et al., 1978] Algorithm 3 is guaranteed to output a solution $\hat{A}$ such that*

$$f(\hat{A}) \geq (1 - 1/e) f(A^*) \approx 0.63 f(A^*), \tag{1.36}$$

where $A^* \in \text{argmax}_{|A|\leq k, A\subseteq V} f(A)$ is the optimal solution for Problem 2.

Using the notion of curvature κ_f of the submodular function f ($\kappa_f \in [0, 1]$ and see Section 1.4.4 for definition of κ_f), one can refine the analysis and show an improved approximation factor in terms of the curvature κ_f:

Theorem 5. *[Conforti and Cornuejols, 1984] Algorithm 3 is guaranteed to output a solution $\hat{A}$ such that*

$$f(\hat{A}) \geq \frac{1}{\kappa_f}(1 - e^{-\kappa_f}) f(A^*) \geq (1 - 1/e) f(A^*). \tag{1.37}$$

We remark that the bound recovers $(1-1/e)$ when the submodular function is fully curved, i.e., $\kappa_f = 1$. The guarantee is improved when the curvature κ_f satisfies that $\kappa_f < 1$. The bound also implies that the greedy algorithm exactly solves Problem 2 when the submodular function f is modular, i.e., $\kappa_f = 0$.

Despite the improved analysis in terms of κ_f, the greedy algorithm still attains a guarantee of $(1 - 1/e)$ when f is fully curved. Therefore, a critical question remains: "Is there any polynomial time algorithm that is guaranteed to achieve a solution with higher approximation factor?" The answer to this question is negative as proved in the following Theorem.

Theorem 6 ([Feige, 1998])**.** *There does not exist any polynomial time algorithm that solves Problem 2 with an approximation factor better than* $(1 - 1/e)$ *unless* $P = NP$.

It implies that the greedy algorithm, although simple, already achieves the best possible approximation guarantee for Problem 2.

The time complexity of the greedy algorithm is $O(nk)$ function evaluations. When the constraint k is in the order of n, the complexity becomes quadratic in the problem size, which is computationally infeasible for large-scale applications. Fortunately, submodularity can be utilized to scale up the greedy implementation leading to an algorithm often known as *lazy greedy* (LazyGreed) [Minoux, 1978]. The key insight here is that the marginal gain of any element $v \in V$ is non-increasing during the greedy algorithm (a consequence of the submodularity of f). Instead of recomputing $f(a|A)$ for each $a \in V \setminus A$, LazyGreed maintains a list of upper bounds $\rho(a)$ on each item's current marginal gain. They are initialized as $\rho(a) \leftarrow f(a)$, and sorted in decreasing order. In iteration i, the algorithm pops the element a off the top of the priority queue and updates the bound $\rho(a) \leftarrow f(a|A)$. a is selected if $\rho(a) \geq \rho(v)$, where v is at the current top of the priority queue, since submodularity in such case guarantees that a provides the maximal marginal gain. Otherwise, we appropriately place the updated $\rho(a)$ back in the priority queue and repeat. Although LazyGreed, in the worst case, still requires $O(kn)$ function evaluations, it is often, in practice, much more efficient than the naïve greedy implementation (Alg. 3). We empirically often observe that

only $O(n)$ function evaluations is needed for LazyGreed on the submodular functions that we utilize in the experiments.

Recently, there has been a number of further advances in scaling up LazyGreed to even larger problem sizes. Parallel computing approaches are often a natural pursuit for solving large-scale algorithmic challenges, and some instances of distributed algorithms for submodular optimization have already been investigated for Problem 2 on extremely large-scale data. For example, [Chierichetti et al., 2010] propose a distributed algorithm to solve Problem 2 with the set cover function as the objective, and with an approximation factor of $(1-1/e-\epsilon)$. Similarly, [Kumar et al., 2013] propose a distributed algorithm to solve Problem 2 with any submodular objective and with an approximation factor of $(1/2-\epsilon)$. Motivated by the difficulty of rendering the entire data set centrally for function evaluation, [Mirzasoleiman et al., 2013] propose a two-stage algorithmic framework to solve Problem 2 with an approximation factor of $O(\frac{1}{\min(k,m)})$ where k and m are the cardinality constraint and the number of distributed partitions, respectively. Later, [Barbosa et al., 2015] show that the same distributed algorithm is guaranteed to achieve a constant factor of $(\frac{1-1/e}{2})$ on expectation if the data is partitioned in a uniformly random manner. All these distributed algorithms can be implemented in a Map-Reduce style.

If restricted to uni-processor approach, existing efficient algorithms for Problem 2, in general, focus on scaling up LazyGreed in one or more of the three directions: (1) reducing the number of function evaluations required for the algorithm [Badanidiyuru and Vondrák, 2014, Mirzasoleiman et al., 2015], (2) decreasing the complexity of function evaluations by using simpler surrogate (proxy) functions [Iyer et al., 2013b, Lindgren et al.,], (3) reducing the memory footprint [Mei et al., 2016]. As we will see in Chapter 6, we propose a multi-stage framework (MultGreed) that directly addresses the time and memory complexity issues of LazyGreed in all three aforementioned directions.

To reduce the number of function valuations needed, [Badanidiyuru and Vondrák, 2014] propose a greedy procedure based on a decreasing threshold, which is guaranteed to yield a $(1-1/e-\epsilon)$-approximation with only $O(\frac{n}{\epsilon}\log\frac{n}{\epsilon})$ function valuations. Similarly, a lazier than

lazy greedy algorithm is given by [Mirzasoleiman et al., 2015] to always find a solution with an approximation factor $(1 - 1/e - \epsilon)$ in expectation in $O(n \log \frac{1}{\epsilon})$ function calls. Instead of searching among V for the item with the maximum marginal gain in each iteration as LazyGreed, this algorithm uniformly and independently samples a random subset of items, from which it searches for the item with the maximum marginal gain. In [Iyer et al., 2013b, Lindgren et al.,], several classes of simpler surrogate functions are proposed to reduce the time and memory complexity of evaluating certain submodular functions, hence scaling the optimization framework up to larger problem sizes. On the other hand, it is also very desirable to prune out items of the ground set significantly (hence reducing the memory footprint) without incurring a significant performance loss. This is the essential idea behind the work on the reducibility of submodular functions [Mei et al., 2016].

Another line of work independent of the three aforementioned directions focuses on maximizing a submodular function under the streaming setting [Badanidiyuru et al., 2014, Bateni et al., 2013, Buchbinder et al., 2015]. For example, a threshold-based procedure is proposed in [Badanidiyuru and Vondrák, 2014], where the algorithm maintains $O(\frac{\log k}{\epsilon})$ candidate solutions simultaneously at any given iteration and performs only one pass over the ground set V with computing the adding gain of each item to all the candidate solutions. Such algorithm requires $O(\frac{k \log k}{\epsilon})$ memory footprint, a total number of $O(\frac{n \log k}{\epsilon})$ function valuations, and is guaranteed to achieve a solution with a factor $1/2 - \epsilon$. A swapping-based algorithmic framework is studied by [Buchbinder et al., 2015]. Instead of maintaining multiple candidate solutions as [Badanidiyuru and Vondrák, 2014], the swapping-based approach only maintains one candidate solution at any given iteration. With only one pass of the data, the algorithm, at each iteration, swap the item into the candidate solution if sufficient gain in the function valuation can be achieved by such operation. With an appropriately chosen threshold scheme, this algorithm is guaranteed to output a solution with a factor $1/4$. This algorithm needs to evaluate the function nk times and a memory complexity of only k. Furthermore, [Buchbinder et al., 2015] show that it is information theoretically hard for any deterministic streaming algorithm to achieve a performance guarantee better than

1/2. Another streaming algorithm is proposed by [Bateni et al., 2013], where the algorithm is simply adapted from the algorithm for the k-secretary problem. Assuming a uniformly random ordering of the data stream with a known length n, the algorithm first partitions the data stream into k equally sized blocks, and runs a standard secretary algorithm to pick an element with the maximum marginal gain conditioned on the current solution in each block. Such algorithm also takes only one pass of the data, n function valuations, and a memory of size k. [Bateni et al., 2013] show that a constant factor ($\frac{1-1/e}{7}$) is achievable on expectation.

Algorithm 4: Greedy algorithm for submodular maximization under a matroid constraint.

Input: A submodular function $f : V \to \mathbb{R}_+$ and a matroid $\mathcal{M}(V, \mathcal{I})$.
Initialize: $A \leftarrow \emptyset$.
while $\exists a \in V \setminus A : A \cup \{a\} \in \mathcal{I}$ **do**
    $a \in \operatorname{argmax}_{a \in V \setminus A : A \cup \{a\} \in \mathcal{I}} f(a|A)$.
    $A \leftarrow A \cup \{a\}$.
Output: $\hat{A} \leftarrow A$.

In the following, we study a slightly more general form of a constrained submodular maximization problem. Here, we consider the matroid constraint (see Section 1.5.3 for definition):

Problem 3 (Matroid constrained submodular maximization)**.**

$$\max_{A \in \mathcal{I}(\mathcal{M})} f(A), \tag{1.38}$$

where $\mathcal{M}(V, \mathcal{I})$ is a matroid.

Note that Problem 2 is a special instance of Problem 3 with a k-uniform matroid. Hence, Problem 3 captures a larger classes of optimization problems than Problem 2. Similar to Problem 2, the simple greedy algorithm also solves Problem 3 with a constant factor. We

describe the greedy algorithm in Alg. 4. Note that Alg. 4 generalizes the Alg. 1 for solving the modular version of Problem 3. The idea here is to keep adding an item to the solution if the maximum marginal gain is achieved while the solution after the update is still independent. Despite its simplicity, 1/2-approximation guarantee can be shown for this algorithm.

Theorem 7 ([Fisher et al., 1978]). *Algorithm 4 is guaranteed to find a solution $\hat{A}$ such that*

$$f(\hat{A}) \geq \frac{1}{2} f(A^*), \tag{1.39}$$

where $A^* \in \operatorname{argmax}_{A \in \mathcal{I}(\mathcal{M})} f(A)$.

We point out that a simple generalization of Problem 3 is to consider more than one matroid as the constraint. Given p different matroids, Problem 3 may be generalized as maximizing f under the constraint that the solution is independent for all p matroids[1]. Although the intersection of p matroids, in general, is not a matroid, the same greedy procedure can be slightly modified to solve the generalized problem with a $(\frac{1}{p+1})$-approximation guarantee [Fisher et al., 1978]. Since most of our problem formulations are defined via a single matroid constraint, we, therefore, focus only on Problem 3 here. It is also worth noting that the same lazy evaluation trick for Problem 2 applies here to significantly speed up the greedy implementation leading to an almost linear time complexity algorithm. Similar to Problem 2, the analysis may be refined using the notion of the curvature κ_f of the submodular function f.

Theorem 8 ([Conforti and Cornuejols, 1984]). *Algorithm 4 is guaranteed to output a solution $\hat{A}$ such that*

$$f(\hat{A}) \geq \frac{1}{1+\kappa_f} f(A^*) \geq \frac{1}{2} f(A^*), \tag{1.40}$$

where $\kappa_f \in [0, 1]$ is the total curvature of the submodular function f.

When the submodular function is fully curved, i.e., $\kappa_f = 1$, Theorem 8 recovers the 1/2-approximation bound shown in Theorem 7. When the submodular function f is modular,

[1]The intersection of p matroids is also known as the p-system

i.e., $\kappa_f = 0$, Theorem 7 implies that the optimal solution is found by Alg. 4, hence, recovers a part of Theorem 3. Furthermore, Theorem 8 suggests that better approximation bound can be shown by Alg. 4 as the submodular function f becomes less curved (i.e., with a smaller curvature κ_f).

On the other hand, it is still worth noting that the approximation factor of Alg. 4, in the worst, case, is still 1/2. However, the best known NP-hard hardness for Problem 3 is $(1 - 1/e)$, which follows from the hardness result for Problem 2 proved by [Feige, 1998]. A critical question remains: "Is there any polynomial time algorithm that achieves the optimal guarantee of $(1 - 1/e)$?" This question remains open until recently it is answered positively by [Vondrák, 2008], who give a multi-linear extension-based algorithm that achieves the optimal factor $(1 - 1/e)$.

The idea of the relaxation algorithm proposed in [Vondrák, 2008] is to relax the discrete optimization problem as a continuous problem, where the submodular function f is relaxed as its multi-linear extension $\hat{f}$, and the matroid constraint $\mathcal{M}(V, \mathcal{I})$ is relaxed as a matroid polytope constraint. [Vondrák, 2008] utilizes a continuous greedy procedure to find a fractional solution for the continuous relaxation problem and then round the fractional solution to a vertex of the matroid polytope (an independent set) via the pipage rounding technique. Interestingly, [Vondrák, 2008] show that such a procedure guarantees a factor of $(1 - 1/e)$ for Problem 3. We point out here that the continuous greedy algorithm, although theoretically optimal, is computationally very intensive, due to the high order time complexity for discretizing the continuous greedy procedure, for pipage rounding procedure, and for evaluating the multi-linear extension of a submodular function. A naive implementation of the continuous greedy algorithm requires a time complexity of at least $O(n^7)$ for most commonly used submodular functions [Iyer et al., 2014].

From the practical point of view, we suggest using the simple greedy algorithm as Alg. 4 for solving Problem 3 on real-world large-scale applications. Moreover, most of algorithmic advances in scaling up the greedy algorithm for Problem 2 easily generalize to the greedy algorithm for Problem 3.

Algorithm 5: Greedy algorithm for Problem 4 [Krause and Guestrin, 2005b]

Input: b, V, c, and f.

Initialization: $A \leftarrow \emptyset$.

while $c(A) < b$ **do**

  $a^* \leftarrow \text{argmax}_{a \in V \setminus A : c(a)+c(A) \leq b} \frac{f(a|A)}{c(a)}$.

  $A \leftarrow A \cup a^*$.

$a^* \leftarrow \text{argmax}_{a \in V : c(a) \leq b} f(a)$.

Output: $\hat{A} \leftarrow \text{argmax}\{f(A), f(a^*)\}$.

Next, we study the problem of maximizing a submodular function under a knapsack constraint. This variant of submodular maximization problem, as we will see, is particularly useful in the speech data subset selection problem. We mathematically define the problem as follows:

Problem 4 (Knapsack constrained submodular maximization)**.**

$$\max_{A \subseteq V, c(A) \leq b} f(A), \tag{1.41}$$

where $c : V \rightarrow \mathbb{R}$ is a modular function ($c(A) \triangleq \sum_{a \in A} c(a), \forall a \in A$) with $c(a)$ measuring the cost of each item $a \in V$, and b is the budget.

Problem 4 generalizes Problem 2 when $c(a) = 1, \forall a \in V$ and $b = k$. On the other hand, Problem 4 generalizes the well-known knapsack problem when the submodular function f is modular. Since the knapsack constraint cannot be captured by any matroid, Problem 4 defines a different class of optimization problems from Problem 3. Interestingly, almost the same greedy heuristic can be used to near-optimally solve Problem 4. We describe the greedy algorithm in Alg. 5. Note that the variant of the greedy algorithm here differs from the greedy algorithm for the cardinality constraint case in two ways:

1. The item with the maximum marginal gain normalized by its cost is chosen in each iteration,

2. A final comparison of the solution to the feasible singleton item with the maximum function value is added.

The first modification is natural in that the cost of selecting an item is accounted for in the algorithm. The second modification, on the other hand, is needed for preserving a constant approximation bound. In practice, the last step may not be necessary, since it is often that the singleton value of any item $a \in V$ is much smaller than the solution $f(A)$ when the budget b is reasonably large. The algorithm attains an approximation factor of $\frac{1}{2}(1-1/e)$ as shown below:

Theorem 9 ([Krause and Guestrin, 2005b])**.** *Algorithm 5 is guaranteed to output a solution* $\hat{A}$ *such that*

$$f(\hat{A}) \geq \frac{1}{2}(1-1/e)f(A^*), \tag{1.42}$$

where $A^* \in \text{argmax}_{A \subseteq V : c(A) \leq b} f(A)$.

The same lazy evaluation trick proposed in [Minoux, 1978] also applies here to scale up the greedy implementation, again, leading to an almost linear time algorithm in practice. Since Problem 4 generalizes Problem 2, the hardness analysis of $(1-1/e)$ for Problem 2 also applies here. The greedy procedure, though attains a constant factor approximation, has not yet achieved the optimal guarantee. [Sviridenko, 2004] show that the optimal guarantee of $(1-1/e)$ is possible by modifying Alg. 5. The modified algorithm enumerates over all feasible subsets of cardinality three, runs an independent instance of Alg. 5 with each subset as the initialization set, and outputs the best solution among all instances. Although theoretically tight, this algorithm requires $O(n^5)$ function valuations, hence is not feasible for large-scale applications. Often, the simple and efficient greedy algorithm as Alg. 5 suffices to yield a very nice solution for Problem 4.

Several generalized forms of Problem 4 have been considered recently by a number of authors [Iyer and Bilmes, 2013, Kulik et al., 2009]. For example, [Iyer and Bilmes, 2013] generalize the knapsack constraint to a submodular cost constraint, where the cost function

c is assumed to be submodular instead of modular as Problem 4. They show that the generalized problem is much harder and is information theoretically hard to approximate within a factor $O(\frac{1}{\sqrt{n}})$. They also give an approximation algorithm with a guarantee of $O(\frac{1}{\sqrt{n}\log n})$ matching the lower bound up to a log factor. Another extension of Problem 4 is to consider multiple knapsack constraints as is done in [Kulik et al., 2009], where they give a scheme achieving an approximation factor $(1 - 1/e - \epsilon)$.

We conclude this section by surveying the algorithms for the constrained case of maximizing a non-monotone submodular function, for which several approximation algorithms are designed for various choices of constraints. The first constant factor approximation algorithm for maximizing a non-monotone submodular function under the intersection of k matroids (with a guarantee of $\frac{1}{2+k+\frac{1}{k}}$), or k knapsack constraints (with a guarantee of $\frac{1}{5}$) is given by [Lee et al., 2009]. [Vondrák, 2013] study the case where the optimized set is a base of a matroid and give an approximation algorithm with a factor 0.309. This was later improved to 0.325 by [Gharan and Vondrák, 2011] using the simulated annealing approach. It was later further improved to $(1/e)$ by [Feldman et al., 2011]. Recently, an improved approximation algorithm has been designed by [Buchbinder et al., 2014], which breaks the $(1/e)$ for the k-uniform matroid case.

1.5.6 Other combinatorial optimization problems involving submodularity

In this section, we study several instances of combinatorial optimization problems that involve submodular functions as the objective. A class of optimization problems that has recently drawn more attention is the difference of submodular (DS) optimization problem as defined below:

Problem 5 (Difference of submodular optimization)**.**

$$\max_{A \in \mathcal{C}} f_1(A) - f_2(A), \tag{1.43}$$

where f_1 and f_2 are monotone submodular functions and $\mathcal{C}$ defines the feasibility sets.

Here $\mathcal{C}$ may be any constraint such as non constraint (i.e., $\mathcal{C} = \{A \subseteq V\}$), the cardinality constraint, the matroid constraint, or the knapsack constraint. The maximization problem is equivalent to the minimization case with $f_2 - f_1$ as the objective. The DS optimization formulation captures a very general class of combinatorial optimization problems as a result of the fact that any general set function can be decomposed as a difference of submodular functions [Narasimhan and Bilmes, 2005]. On the other hand, a non-trivial such construction may require an exponential time complexity in the worst case. Moreover, [Iyer and Bilmes, 2012] show that Problem 5 is extremely hard: it is information theoretically hard to approximate within any polynomial function of the problem size. In other words, such optimization problem is inapproximable to any factor. Despite the inapproximability, DS optimization has still found a number of real-world applications [Iyer and Bilmes, 2012,Bach, 2011,Narasimhan and Bilmes, 2005,Kawahara et al., 2011].

The general idea for optimizing a DS function is to utilize an iterative procedure, where, in each iteration, the underlying objective is approximated by a tight submodular surrogate defined via the semigradients of both submodular functions f_1 and f_2. A common approach for Problem 5 is to, in each iteration, approximate f_2 with a tight modular lower bound leading to a subproblem in the form of (constrained) non-monotone submodular maximization, which there exists a vast number of efficient and near-optimal algorithms.

If any general set function can be naturally written as a difference of submodular functions f_1 and f_2, this paradigm provides a principled approach for applying the vast literature on submodular optimization to solve the DS optimization. Despite the lack of approximation factor, such paradigm guarantees that the solution is improved in successive iterations and is bounded with an additive error to the optimal solution [Iyer and Bilmes, 2012]. As we will see in Chapter 5, Problem 5 naturally occurs as the optimization formulation for the problem of interactive learning of submodular functions.

Next we focus on another class of optimization problems that involve submodularity. In this case, the goal is to minimize the ratio of submodular (RS) functions as defined below:

Problem 6 (Ratio of submodular functions minimization)**.**

$$\min_{\emptyset \subset A \subseteq V} \frac{f_1(A)}{f_2(A)}, \tag{1.44}$$

where f_1 and f_2 are monotone submodular functions, where $f_1(A) > 0, f_2(A) > 0, \forall \emptyset \subset A \subseteq V$.

Problem 6 has recently been systematically studied in [Bai et al., 2016]. They show nice connections of RS minimization to Problem 5. Special cases of Problem 6 have been studied with either f_1 and/or f_2 being modular. Interestingly, when f_2 is modular, Problem 6 can be reduced to submodular minimization, which is polynomial time solvable. When f_1 is modular, the problem can be near-optimally solved with a greedy procedure with a factor $(1 - 1/e)$. For the most general case (i.e., f_1 and f_2 are submodular), [Bai et al., 2016] show that the problem is information theoretically hard to approximate better than $O(\sqrt{n})$. Furthermore, they give an ellipsoidal approximation based algorithm that attains a factor of $O(\sqrt{n} \log n)$ matching the hardness up to a log factor.

For the completeness of the survey, we also briefly introduce the submodular partitioning problems here. We defer the readers to Chapter 7 for the detailed discussions as well as efficient optimization algorithms for the submodular partitioning problems. Here we formalize the mathematical formulation of the problems introduced in Section 1.2. Recall that given V as the ground set, we define the set of sets $\pi = (A_1^\pi, \ldots, A_m^\pi)$ as the partitioning of the ground set V into m blocks. We also define Π as the set of all partitioning of V into m blocks. Note that π satisfies the following conditions:

- Union gives the ground set: $\cup_{i=1}^m A_i^\pi = V$.
- Empty intersection: $A_i^\pi \cap A_j^\pi = \emptyset, \forall i \neq j$.

Given m monotone submodular functions: $f_1, \ldots, f_m$ that valuates the utility of the data sets, a natural optimization problem is given below:

Problem 7 (Max-(Min+Avg) Submodular Partitioning)**.**

$$\max_{\pi\in\Pi}\Big[\bar{\lambda}\min_i f_i(A_i^\pi)+\frac{\lambda}{m}\sum_{j=1}^{m} f_j(A_j^\pi)\Big], \tag{1.45}$$

where $0\leq\lambda\leq 1$, $\bar{\lambda}\triangleq 1-\lambda$.

On the other hand, when the monotone submodular functions $f_1,\ldots,f_m$ measure the notion of cost and complexity, another natural formulation is as follows:

Problem 8 (Min-(Max+Avg) Submodular Partitioning)**.**

$$\min_{\pi\in\Pi}\Big[\bar{\lambda}\max_i f_i(A_i^\pi)+\frac{\lambda}{m}\sum_{j=1}^{m} f_j(A_j^\pi)\Big], \tag{1.46}$$

where $0\leq\lambda\leq 1$, $\bar{\lambda}\triangleq 1-\lambda$.

In both problems, the parameter λ controls the objective: $\lambda=1$ is the average case, $\lambda=0$ is the robust case, and $0<\lambda<1$ is a mixed case. By definition, Problem 7 asks for a partition whose blocks each (and that collectively) are a good, say, summary of the whole. Problem 8, on the other hand, asks for a partition whose blocks each (and that collectively) are internally homogeneous (as is typical in clustering). Taken together, we call Problems 7 and 8 *Submodular Partitioning*. We further categorize these problems depending on if the f_i's are identical to each other (*homogeneous*) or not (*heterogeneous*). The heterogeneous case clearly generalizes the homogeneous setting, but as we will see in Chapter 7, the additional homogeneous structure can be exploited to provide more efficient and/or tighter algorithms.

These problems are, in fact, very general. They generalize purely robust instances of the problem, namely *max-min submodular fair allocation* (Problem 7 with $\lambda=0$) [Golovin, 2005] and *min-max submodular load balancing* (Problem 8 with $\lambda=0$) [Svitkina and Fleischer, 2008], and also average-case instances, that is the *submodular welfare problem* (Problem 7 with $\lambda=1$) [Vondrák, 2008] and *submodular multiway partition* (Problem 8 with $\lambda=1$) [Chekuri and Ene, 2011a].

While the robust versions have been studied in the theory community [Goemans et al., 2009, Golovin, 2005, Khot and Ponnuswami, 2007, Svitkina and Fleischer, 2008, Vondrák,

2008], existing work has focused on tight approximation guarantees, and the resultant algorithms are not generally scalable to large real-world applications. This is in contrast to the average case, where most of the algorithms are scalable. As we will see in Chapter 7, we bridge this gap, by proposing several new algorithms (including greedy, majorization-minimization, minorization-maximization, and relaxation algorithms) that not only scale to large datasets but that also achieve theoretical approximation guarantees comparable to the state-of-the-art. We moreover provide new scalable algorithms that apply to additive combinations of the robust and average-case objectives.

Part I
DATA SUMMARIZATION

In the first part of this thesis, we focus on the problem of data summarization. We propose a generic data summarization paradigm based on submodular function optimization. Our main focus is on both the modeling and the optimization components of the submodular data summarization paradigm. We examine the efficacy of submodular utility model on three concrete case studies, including speech data subset selection (Chapter 2), genomics assay panel selection (Chapter 3), and batch active learning (Chapter 4). As we will see, appropriate submodular utility functions naturally occur as the model for these tasks. To further strengthen the expressive power of submodular utility functions, we propose a learning framework that "interactively" learns the mixtures of submodular function (Chapter 5). Lastly, we address the efficiency of the submodular data summarization paradigm by proposing a novel multi-stage algorithmic framework to significantly scale up the paradigm while attaining comparable optimality guarantees (Chapter 6).

Chapter 2

CASE STUDY I: SPEECH DATA SUBSET SELECTION

Present-day automatic speech recognition (ASR) systems are trained on vast amounts of acoustic data. Although larger training data sets often lead to gains in system performance, there are well-known problems associated with ever-increasing data sets:

1. Larger data sets place greater demands on available resources, such as storage and CPU cycles.

2. Existing software infrastructure often needs to be modified to be able to process ever-larger data sets, which requires developer time and expertise.

3. An aggravating factor is that many statistical learning procedures (e.g., the Expectation-Maximization algorithm typically used for training Gaussian Mixtures Models (GMM), or the back-propagation algorithm for training neural networks) typically process training data sets repeatedly.

4. More importantly, the gains in system performance achieved by increasing training data sets are often sublinear: after an initial increase the gains become smaller, a phenomenon known as diminishing returns.

The last issue is due to that additional data may be noisy, irrelevant to the task at hand or, most probably, fully or partially redundant with already existing data. Having unnecessary and redundant data thus results in wasted computational resources. Therefore, it is critical to develop methods to select an informative and representative subset of a large data set that retains as many of the benefits of the large data set as possible, while simultaneously

minimizing resource requirements, a goal that naturally falls under the data summarization paradigm.

Speech data subset selection, in general, has been conducted for several different scenarios that frequently occur in the field of speech processing:

1. Identifying a smaller subset of the data that fits a given budget but provides as much information as the original large data set.

2. Data selection for human annotation, which is of interest when developing ASR systems for new languages or dialects whose audio data has not yet been transcribed.

3. Selection for adaptation, where the goal is to tune a system to a known development or test set.

In this Chapter, we concentrate on the first two scenarios. For the first scenario, we assume that the manual labels of the acoustic data are available to a data subset selection algorithm and call such setting *supervised data subset selection*. For the second scenario, we constrain ourselves to the *unsupervised data subset selection* setting, where the subset selection framework does not assume access to any of the data labels and is solely dependent on the acoustic features of the data. For both the supervised and the unsupervised settings, we are interested in studying the following question: given a drastic reduction in training set size (from one to two orders of magnitude), what is the smallest degree of information loss possible?

In the supervised setting, the intended benefit is to significantly shorten experimental turn-around time – often, systems need to be trained repeatedly with different model configurations or parameters. If this could be done on a small subset, more experimentation could be performed within a given time period, and systems could be more highly and accurately tuned. In the unsupervised setting, the intended benefit is that an equally effective ASR system can be trained using a smaller but wisely selected manually labeled data set than a randomly selected set.

Most approaches to data subset selection in speech recognition are based on the framework of active learning, where additional training data is chosen to update an already existing system [Hakkani-Tur et al., 2002, Lamel et al., 2002, Kemp and Waibel, 1998]. Under this approach the utility of each sample in the training data set is equivalent to the confidence score given by an existing ASR system. All samples are then ranked according to their utility score, and the samples with highest scores are selected. The drawback of this approach is that it requires a fully-trained speech recognizer with reasonable performance that repeatedly iterates over the training set.

In [Liu et al., 2007], data selection at different levels (e.g., utterance level, phone level, and frame-level) is performed for the purpose of discriminative training of acoustic models. The selection criterion is the average phone accuracy of utterances. In [Itoh et al., 2012], they propose a method for data subset selection that meets the criteria of both informativeness and representativeness. The informative score of an utterance is computed as the entropy of its N-best word hypotheses produced by a baseline ASR system. The representative score of an utterance with respect to a data pool is calculated as the average TF-IDF similarity with all other utterances in the pool. Like the active selection methods, this approach requires a word recognizer.

A number of authors study selecting acoustic data based on the coverage of the linguistic units (e.g., phonemes, words, triphones, and etc.) as the selection criterion. For example, [Wu et al., 2007] propose a method to subselect acoustic training data based on the transcriptions of the data. Their objective is to select a subset of the data that results in a maximum-entropy distribution (or equivalently the most uniform distribution) over linguistic units in the set. On the other hand, [Siohan, 2014] argue that better performance is achievable for semi-supervised update of acoustic models when the acoustic training data is selected such that the resulting set is close in distribution of the linguistic units to that of the a curated development set. As we will see, either criterion is, in fact, easily formulated as an instance of submodular data summarization paradigm.

While the above methods select data using a greedy algorithm, in general there is not a guarantee that such an algorithm applied to the above objectives has any quality assurance. Another line of work focuses on formulating the data selection as a constrained submodular maximization problem. Submodular data selection was first presented in [Lin and Bilmes, 2009] for a small-scale phonetic recognition task on TIMIT under a supervised scenario. A KL-divergence submodular function is proposed in [Shinohara, 2014] and shown to be submodular. Given any desirable distribution, the utility of a data set is modeled as how close the distribution of the speech units within the selected data set is to the desirable distribution measured by KL-divergence. However, no empirical evaluation is given in [Shinohara, 2014] to assess the performance of this objective.

As we will see, the presented results in this Chapter significantly extends this line of research. We investigate both the supervised and unsupervised settings for the small scale phone recognition task evaluated on TIMIT. We, moreover, propose a general and scalable feature-based function and show the connection of the proposed framework to the data selection criteria in [Siohan, 2014], [Wu et al., 2007] and [Shinohara, 2014]. We lastly empirically demonstrate the success of the proposed framework on a large-vocabulary speech recognition task.

2.1 Submodular Data Subset Selection Framework

2.1.1 Problem Formulation

Suppose we have a set of speech utterances $V = \{v_1, v_2, \ldots, v_n\}$. Consider a monotonically non-decreasing submodular set function $f : 2^V \rightarrow \mathbb{R}$, which maps each subset $A \subseteq V$ to a real number that represents the value $f(A)$ of subset A. The speech data subset selection problem, then, can be viewed as maximizing the value $f(A)$ of A such that the cost of the selected subset A does not exceed a given budget. Here, we use a modular function $c : V \rightarrow \mathbb{R}$ to define the cost of each speech utterance. We denote $c(v)$ as the cost of the utterance v. Therefore, the cost of a subset A is simply the sum of the individual cost,

namely $c(A) = \sum_{a \in A} c(a)$. In this work, we define the cost $c(v)$ of a speech utterance v as its length. Mathematically, the problem can be formulated follows:

$$\max_{A \subseteq V, c(S) \leq b} f(A) \tag{2.1}$$

where b is a budget on the amount (or cost) of speech data to be selected and $c(A)$ measures the amount (or cost) of speech contained in a subset A of the whole corpus. Note that the combinatorial optimization problem here asks for maximizing a monotone submodular function under a knapsack constraint (Problem 4). The scalable and efficient greedy algorithm (Alg. 5) solves the problem with a constant factor $\frac{1}{2}(1 - 1/e)$. We defer the readers to Section 1.5.5 for more details about this optimization problem.

2.1.2 Submodular Objectives

Having defined the formulation, the remaining problem is how to appropriately choose the submodular objective f that best characterizes the utility of the acoustic data for training ASR systems. To this end, we introduce a broad range of submodular functions that have shown success on this task. We first introduce two classes of submodular functions that are defined via a pairwise similarity graph. The first objective is called the *facility location* function with the following form:

$$f_{\text{fac}}(A) = \sum_{i \in V} \max_{j \in A} w_{i,j}, \tag{2.2}$$

where $w_{i,j} \geq 0$ indicates the similarity between utterances i and j. For now, we assume the similarity between utterances is given. In Sec 2.1.3, we formally introduce a class of methods to characterize the similarity between utterances that accounts for both the acoustic and phonetic information.

Sharing a similar flavor, *Saturated coverage* functions also naturally models the notion of representativeness and diversity in a set of speech utterances. Successfully applied in

document summarization tasks [Lin and Bilmes, 2011], the function has the following form:

$$f_{\text{sat}}(A) = \sum_{i \in V} \min\{C_i(A), \alpha C_i(V)\} \tag{2.3}$$

where $C_i(A) = \sum_{j \in A} w_{ij}$ and $0 \leq \alpha \leq 1$ is the saturation coefficient. Note that $C_i : 2^V \rightarrow \mathbb{R}$ is itself monotone submodular (modular in fact).

Both the facility location function and the saturated coverage funcions are graph-based, since a pairwise similarity graph is required to instantiate them. Even with highly optimized data structures, efficient computation of similarity measures, and graph approximations, graph construction can become computationally prohibitive when $|V|$ is big (e.g., millions, or larger).

An alternative class of submodular functions that do not require a pair-wise similarity graph is *feature-based*, defined as:

$$f_{\text{fea}}(A) = \sum_{u \in U} w_u g(m_u(A)) \tag{2.4}$$

where $g()$ is a non-negative monotone non-decreasing concave function (e.g., $g(x) = \sqrt{x}$ and $g(x) = \log(x)$), U is a set of features, w_u is the weight for the feature u, the modular feature function $m_u(A) = \sum_{j \in A} m_u(j)$ is a non-negative score for feature u in a set A, with $m_u(j)$ measuring the degree of feature u present in utterance $j \in A$. In the context of speech data subset selection, U can take various forms including triphones, words, phonemes, acoustically derived measures, etc. Moreover, there are many different ways to define the relevance score $m_u(s)$. One simple way might be to define it as the count of frames in the utterance s that has the label u.

Next, we are going to connect the feature based function with the form defined above to the KL-divergence minimization of the distribution in the set of speech units U. Given a set of m speech units $U = \{u_1, \ldots, u_m\}$ and any distribution $\pi = (\pi_{u_1}, \ldots, \pi_{u_m})$ (i.e., $\sum_{u \in U} \pi_u = 1$ and $\pi_u \in [0, 1], \forall u \in U$) of the units that we wish to maintain in a data set, consider a variant of the feature-based submodular function with the following form:

$$f^{\pi}_{\text{fea}}(A) = \sum_{u \in U} \pi_u \log m_u(A), \tag{2.5}$$

where the feature weight w_u is defined as π_u for each unit $u \in U$ and the concave function g is chosen as the log function [1]. Let $p_u(A) = \frac{m_u(A)}{\sum_{u' \in U} m_{u'}(A)}$ be the fraction of the frames labeled as u in a set A of utterances. Let

$$D_{KL}(\pi||p(A)) = \sum_{u \in U} \pi_u \log \frac{\pi_u}{p_u(A)} \tag{2.6}$$

be the KL-divergence between the target distribution π and $p(A)$. Next, we establish the connection of the function $f^{\pi}_{\text{fea}}(A)$ to $D_{KL}(\pi||p(A))$ as follows:

$$D_{KL}(\pi||p(A)) \triangleq \sum_{u \in U} \pi_u \log \frac{\pi_u}{p_u(A)} \tag{2.7}$$

$$= -\sum_{u \in U} \pi_u \log p_u(A) + \sum_{u \in U} \pi_u \log \pi_u \tag{2.8}$$

$$= -\sum_{u \in U} \pi_u \log m_u(A) + \sum_{u \in U} \log \sum_{u \in U} m_u(A) + \sum_{u \in U} \pi_u \log \pi_u \tag{2.9}$$

$$= -f^{\pi}_{\text{fea}}(A) + \sum_{u \in U} \log c(A) + \sum_{u \in U} \pi_u \log \pi_u. \tag{2.10}$$

Eqn 2.9 follows from the definition of $p_u(A) = \frac{m_u(A)}{\sum_{u \in U} m_u(A)}$, and Eqn 2.10 holds since $c(A) = \sum_{u \in U} m_u(A)$. As a reminder, $c(A)$, the cost of the set A, is defined as the total number of frames in the set A. Then, the following optimization problem:

$$\min_{S \subseteq V, c(A) = b} D_{KL}(\pi||p(A)) \tag{2.11}$$

can be equivalently solved as

$$\max_{A \subseteq V; c(A) \leq b} f^{\pi}_{\text{fea}}(A), \tag{2.12}$$

since the third term in Eqn 2.10 is a constant, and the second term is also a constant equal to $|U| \log b$ as a result of the equality constraint in Problem 2.11. Given a budget constraint b, maximizing f^{π}_{fea} with respect to a target distribution π is equivalent to finding

[1] To avoid numerical issues, a slightly modified variant of the form $f^{\pi}_{\text{fea}}(A) = \sum_{u \in U} \pi_u \log(m_u(A) + 1)$ can be used in practice, where the counting function $m_u(A)$ is implemented with add-one smoothing for each speech unit $u \in U$.

the minimizer of $D_{KL}(\pi||p(A))$ with the cost of A being equal to b. We remark that our analysis here builds upon the result presented in [Shinohara, 2014]. However, our analysis demonstrates a stronger connection between the feature-based submodular function to that KL-divergence minimization, since the equivalence is shown for even the non-uniform cost scenario (i.e., the knapsack constraint setting), which was not shown in [Shinohara, 2014].

To encourage selecting a data set with uniform coverage over a set of speech units, one may design the target distribution π as the uniform distribution, i.e., $\pi_u = \frac{1}{m}, \forall u \in U$. However, it should be clear that the KL-divergence function can be used for a much more general scenario, since the target distribution π can be instantiated as any distribution. [Siohan, 2014] reported that a training data set that has a distribution of the triphones matching with that of a development data set yields better performance for semi-supervised training than a randomly sampled data set of similar size. They employ a greedy heuristic to optimize over $D_{\mathrm{KL}}(\pi||p(A))$, where π is the distribution of the triphones in a predefined development data set. However, the greedy heuristic is limited for choosing data sets of any desirable size, and engineering tricks are used for constructing data sets of different sizes. With the submodularity of f^{π}_{fea} and the established connection between f^{π}_{fea} and $D_{\mathrm{KL}}(\pi||p(A))$, one may formulate the data selection criterion proposed by [Siohan, 2014] as Problem 2.1 which can be efficiently solved with favorable optimality guarantees for any selection sizes.

As a slight digression, we next analyze the maximum-entropy criterion proposed in [Wu et al., 2007] for speech data subset selection, where they model the utility of a data set as the level of uniform coverage of a data set A over the speech units U through the notion of entropy:

$$h(A) = -\sum_{u \in U} p_u(A) \log p_u(A). \tag{2.13}$$

Given the defined objective $h(A)$, [Wu et al., 2007] propose to perform data selection by maximizing this objective via a streaming greedy heuristic. Setting an adding threshold τ, the streaming heuristic performs one pass over the entire data and adds an utterance to the selected set if the improvement to the objective h by adding this item is larger than the

threshold τ. The streaming heuristic, though efficient, needs to tune the threshold parameter τ so that data selection of different sizes can be obtained. In some cases, large sizes of data selection (e.g. 10%, or higher) cannot be achieved with this heuristic no matter how the parameter τ is tuned due to saturation of the objective h, as we will see in our experiments (Section 2.2.2). Furthermore, the streaming heuristic does not have any optimality guarantee in terms of the objective being optimized.

As we will see next, this heuristic can be significantly improved leading to a much more principled optimization method thanks to submodularity. For ease of analysis, we first introduce an objective that we call *histogram-entropy* function in the following form:

$$f_{\text{entr}}(A) = -\sum_{u \in U} m_u(A) \log m_u(A). \ ^{2} \tag{2.14}$$

Recall that $m_u(A)$ always takes integer values in this application, since it is defined as the frame count of the speech unit u in the set of utterances A. By definition, f_{entr} is always negative and is, in form, a sum of monotone-decreasing concave functions composed with modular functions, which can be again easily shown to be monotone non-increasing submodular [Stobbe and Krause, 2010]. We will first derive the connection of the objective $h(A)$ to $f_{\text{entr}}(A)$, and, then, show that maximizing $h(A)$ with equal budget constraint is equivalent to optimizing over f_{entr}, and, lastly, describe a method for optimizing a monotone non-increasing submodular function f_{entr}. The connection of $h(A)$ to $f_{\text{entr}}(A)$ can be seen as follows:

$$h(A) = -\sum_{u \in U} \frac{m_u(A)}{\sum_{u' \in U} m_{u'}(A)} \log \frac{m_u(A)}{\sum_{u' \in U} m_{u'}(A)} \tag{2.15}$$

$$= -\sum_{u \in U} \frac{m_u(A)}{c(A)} \log m_u(A) + \log c(A) \tag{2.16}$$

$$= \frac{f_{\text{entr}}(A)}{c(A)} + \log c(A). \tag{2.17}$$

---

[2]Similar to $f^{\pi}_{\text{fea}}(A)$, a slightly modified variant of the form $f_{\text{entr}}(A) = -\sum_{u \in U} m_u(A) \log(m_u(A) + 1)$ can be used in practice to avoid numerial issues, where the counting function $m_u(A)$ is implemented with add-one smoothing for each speech unit $u \in U$.

Then, it's easy to verify that the following optimization problem:

$$\max_{c(A)=b; A\subseteq V} h(A) \tag{2.18}$$

is equivalent to the following:

$$\max_{c(A)\geq b; A\subseteq V} f_{\text{entr}}(A). \tag{2.19}$$

Problem 2.19 does not admit the same form as Problem 2.1 in that $f_{\text{entr}}(A)$ is not monotone non-decreasing, and the inequality in the constraint is reversed. To address this issue, we define an auxiliary function for f_{entr} as $\tilde{f}_{\text{entr}}(A) = f_{\text{entr}}(V \setminus A) - f_{\text{entr}}(V)$. Note that $\tilde{f}_{\text{entr}}$ is normalized monotone non-decreasing submodular. Problem 2.19 can be transformed into an equivalent form as:

$$\max_{c(A)\leq c(V)-b} \tilde{f}_{\text{entr}}(A), \tag{2.20}$$

which is in the form of Problem 2.1, hence, can be solved by the greedy algorithm described in Alg. 5. We take the complement of the output data set by the greedy algorithm as the final output for Problem 2.19. To summarize, the optimization recipe is as follows: (1) Derive the auxiliary function $\tilde{f}_{\text{entr}}$ for f_{entr}. (2) Run Alg. 5 with the input objective as $\tilde{f}_{\text{entr}}$ and the selection budget as $c(V) - b$. (3) Given the solution $\hat{A}$ from Step 2, output its complement $V \setminus \hat{A}$ as the final solution.

Given the above analysis, interestingly enough, we show that the maximum-entropy criterion proposed by [Wu et al., 2007] also admits an equivalent transformation to the constrained submodular maximization formulation, for which one can efficiently and near-optimally optimize.

We next concentrate back to the feature-based submodular function. Although it has shown great flexibility in modeling the utility of data set, one issue with the feature-based functions is that they represent interactions between different items in the whole set V, but cannot represent interactions between different features or sets of features, meaning that information within one feature $u \in U$ might be partially redundant with another feature

$u' \in U$, $u' \neq u$. A solution to this issue is to use a novel construct we call a *two-layer feature-based submodular functions*. Let U^1 be a set of features, U^2 be a set of meta-features, where $|U^1| = d_1, |U^2| = d_2$ and $d_1 > d_2$. Between U^1 and U^2, we define a weight matrix W of dimension $d_2 \times d_1$. Entries in W measure the interactions between the lower-level features in U^1 and corresponding meta-features in U^2. The two-layer feature-based submodular function takes the following form:

$$f_{\text{2-fea}}(A) = \sum_{u_2 \in U^2} g_1\big(\sum_{u_1 \in U^1} W(u_2, u_1) g_2(m_{u_1}(A))\big) \tag{2.21}$$

where $g_1()$ and $g_2()$ are non-negative monotonically non-decreasing concave functions, $m_{u_1}(A)$ takes the same form and interpretation as in the feature-based submodular function. $W(u_2, u_1)$ is the entry in the weight matrix W that measures the interaction between the feature $u_1 \in U^1$ and the feature $u_2 \in U^2$. The submodularity of $f_{\text{2-fea}}(A)$ follows from Theorem 1 in [Lin and Bilmes, 2011].

2.1.3 Similarity Between Speech Utterances

As mentioned in the previous part, some graph-based submodular functions, e.g., facility location function and the saturated coverage function, need to be instantiated on a pairwise similarity graph, which requires computation of similarity measure between any pair of speech utterances. In this section, we describe three methods to characterize the similarity between two speech utterances: (1) Firsher kernel; (2) TF-IDF kernel; and (3) String kernel.

First, we introduce the Fisher kernel to derive the similarity between utterances, which was first presented and used in [Lin et al., 2009]. The *Fisher kernel* is computed based on the vector of derivatives U_X of the log-likelihood of the acoustic data (X) with respect to the parameters in the phone HMMs $\theta_1, ..., \theta_m$ for m models, having similarity score:

$$w_{ij} = (\max_{i',j'} d_{i'j'}) - d_{ij}, \text{ where } d_{ij} = ||U'_i - U'_j||_1,$$

$$U_X^\theta = \bigtriangledown_\theta \log P(X|\theta), \text{ and } U'_X = U_X^{\theta_1} \circ U_X^{\theta_2} \circ \cdots \circ U_X^{\theta_m}.$$

By its definition, Fisher kernel focuses on measuring similarity between utterances in terms of their acoustic features. Essentially, the idea of the Fisher kernel is to first represent a speech utterance of variable length with a fixed length size feature vector. Each entry of the feature vector measures the derivative of the log-likelihood for the corresponding parameter.

Next, we consider similarity measure that moves beyond purely acoustic similarity measures and consider kernels derived from discrete representations of the acoustic signal. First we run a tokenizer over the acoustic signal that converts it into a sequence of discrete labels. In our TIMIT experiments, we use a simple bottom-up monophone recognizer (without higher-level constraints such as a phone language model) that produces phone labels. We then use the hypothesized sequence of phonetic labels to compute two different sentence similarity measures: (a) cosine similarity using TF-IDF weighted phone n-gram counts, and (b) string kernels.

Let s_i be the phone sequence for each speech utterance $i \in V$. We compute the *TF-IDF weighted cosine similarity* between any pair of speech utterances i and j as follows:

$$w_{ij} = \frac{\sum_{w \in s_i} \mathrm{tf}_{w,s_i} \times \mathrm{tf}_{w,s_j} \times \mathrm{idf}_w^2}{\sqrt{\sum_{w \in s_i} \mathrm{tf}_{w,s_i}^2 \, \mathrm{idf}_w^2} \sqrt{\sum_{w \in s_j} \mathrm{tf}_{w,s_j}^2 \, \mathrm{idf}_w^2}}, \tag{2.22}$$

where tf_{w,s_i} is the count of n-gram w in s_i and idf_w is the inverse document count of w (each sentence is a "document").

Many variants of string kernels exist in the literature. In this work, we adopt the gapped, weighted subsequence kernel of the type described in [Rousu and Shawe-Taylor, 2006] as the string kernel. Formally, we define a sentence s as a concatenation of symbols from a finite alphabet Σ (here the inventory of phones) and an embedding function from strings to feature vectors, $\phi : \Sigma^* \to \mathcal{H}$. The string kernel function $\mathcal{K}(s,t)$ computes the distance between the resulting vectors for two sentences s_i and s_j. The embedding function is defined as

$$\phi_u^k(s) := \sum_{\mathbf{i}:u=s(\mathbf{i})} \lambda^{|i|} \quad u \in \Sigma^k \tag{2.23}$$

where k is the maximum length of subsequences, $|i|$ is the length of i, and λ is a penalty

parameter for each gap encountered in the subsequence. $\mathcal{K}$ is defined as

$$\mathcal{K}(s_i, s_j) = \sum_u \langle \phi_u(s_i), \phi_u(s_j) \rangle w_u \tag{2.24}$$

where w is a weight dependent on the length of u, $l(u)$. Finally, the similarity measure is obtained as the normalized the kernel score:

$$w_{i,j} = \frac{\mathcal{K}(s_i, s_j)}{\sqrt{\mathcal{K}(s_i, s_i)\mathcal{K}(s_j, s_j)}}. \tag{2.25}$$

2.2 Experimental Results

We evaluate the proposed speech data subset selection framework on two tasks: (1) small-scale phone recognition task, which is evaluated on the TIMIT corpus; (2) large vocabulary continuous speech recognition (LVCSR) task, which is tested on 1300 hours of conversational English corpus. Both the supervised and unsupervised settings are evaluated on the TIMIT data. We tested only the supervised setting on the large-scale experiments.

2.2.1 Phone Recognition Experiments

Data and Systems

We evaluate our approach on subselecting training data from the TIMIT corpus for training a phone recognizer. Although this is not a large-scale data task, it is an appropriate proof-of-concept task for rapidly testing different combinations of submodular functions and similarity measures. Our goal is to focus on acoustic modeling only; we thus look at phone recognition performance and do not currently take into account potential interactions with a language model. We also chose a simple acoustic model, a monophone HMM recognizer, rather than a more powerful but computationally complex model in order to ensure quick experimental turnaround time. Note that the goal of this study is not to obtain the highest phone accuracy possible; what is important is the relative performance of the different subset selection methods, especially on small data subsets.

The sizes of the training, development, and test data are 4620, 200 and 192 utterances, respectively. Preprocessing was done by extracting 39-dimensional MFCC feature vectors every 10 ms, with a window of 25.6ms. Speaker mean and variance normalization was applied. Following the selection of subsets (1%, 2.5%, 5%, 10%, etc. of the data, measured as percentage of non-silence speech frames), we train a 3-state HMM monophone recognizer for all 48 TIMIT phone classes on the resulting sets and evaluate the performance on the core test set of 192 utterances, collapsing the 48 classes into 39 in line with standard practice. The HMM state output distributions are modeled by diagonal-covariance Gaussian mixtures with the number of Gaussians ranging between 4 and 64, depending on the data size.

We compared our method against two baseline approaches. For a random sampling baseline, we perform 100 random draws of the specified subset sizes and average the results. The second baseline consists of the method in [Wu et al., 2007], where utterances are selected to maximize the entropy of the histogram over phones in the selected subset.

Supervised Data Subset Selection

In the first set of experiments, we test on the supervised setting, where the manual labels of the utterances are available for data subset selection. In this case, we tested the three different similarity measures in combination with the facility location and saturated coverage functions. We first train a 16-component Gaussian mixture monophone HMM system on the full data set. We use the parameters trained from the acoustic model to derive the Fisher kernel similarity. We also utilize the trained system to generate the hypothesized phone sequences for each speech utterance, which are then used the phone sequences for deriving the string kernel and TF-IDF based similarity measures. The parameters of the gapped string kernel (i.e., the kernel order k, the gap penalty λ, and the contiguous substring length l) were optimized on the development set. The best values were $\lambda = 0.1, k = 4, l = 3$.

Figure 2.1 shows the performance of the random and entropy-based baselines as well as the performance of the facility location function with different similarity measures. The entropy-based baseline beats the random baseline for most percentage cases but is otherwise

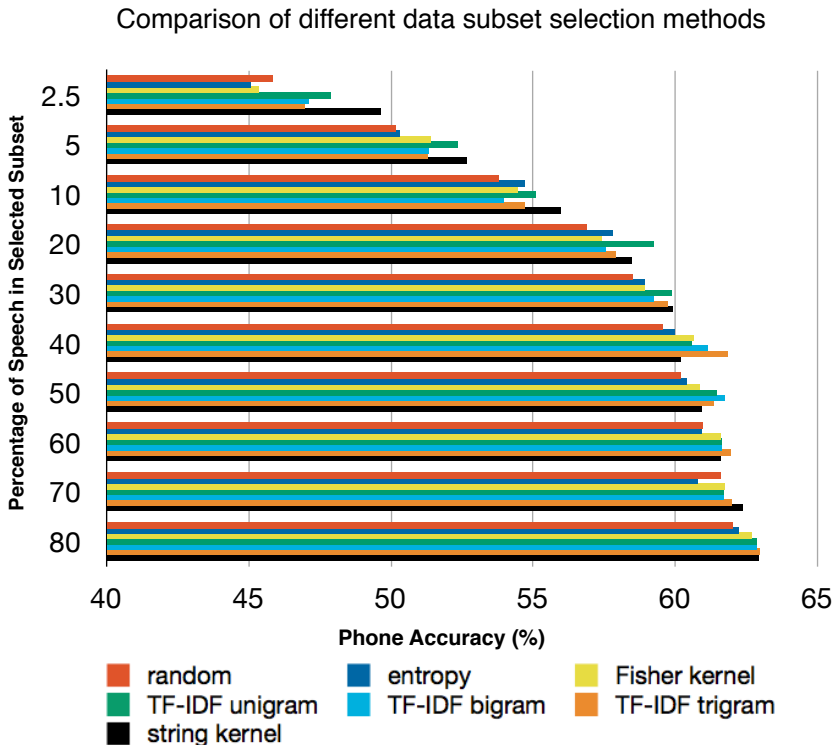


Figure 2.1: Supervised setting: Phone accuracy for different subset sizes; each block of bars lists, from top to bottom: random baseline, entropy baseline, Fisher kernel, TF-IDF (unigram), TF-IDF (bigram), TF-IDF (trigram), string kernel.

the lowest-performing method overall. Note that this baseline uses the true transcriptions in line with [Wu et al., 2007] rather than the hypothesized phone labels output by our recognizer. The low performance and the fact that it is even outperformed by the random baseline in the 2.5% and 70% cases may be because the selection method encourages highly diverse but not very representative subsets. Furthermore, the entropy-based baseline utilizes a non-submodular objective function with a heuristic greedy search method. No optimality guarantee can be made for this method.

Among the different similarity measures, the Fisher kernel outperforms the baseline methods but has lower performance than the TF-IDF kernel and the string kernel. The best

performance is obtained with the string kernel, especially when using small training data sets (2.5%-10%). The submodular selection methods yield significant improvements ($p < 0.05$) over both the random baseline and over the entropy-based method.

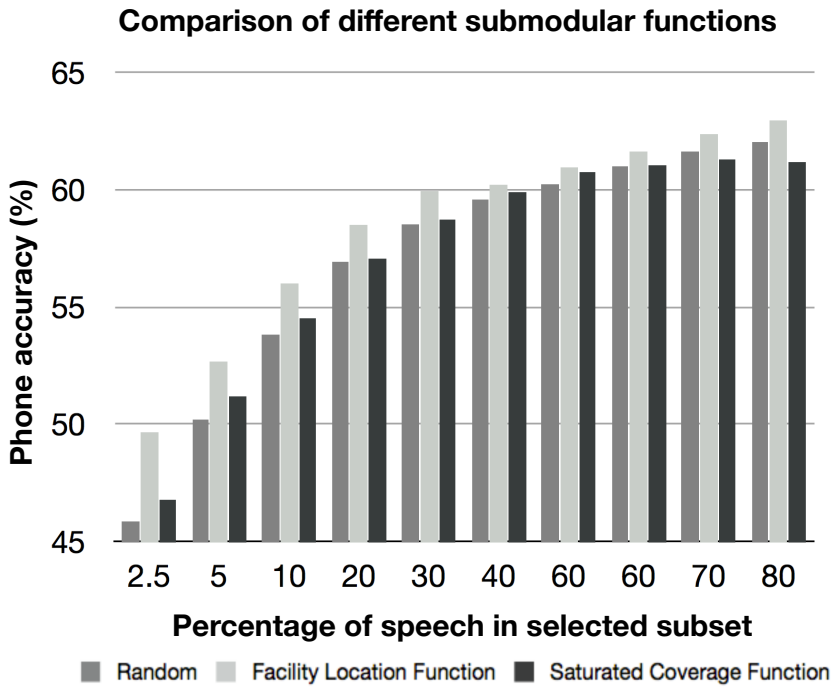


Figure 2.2: Phone accuracy obtained by random selection, facility location function, and saturated coverage function (string kernel similarity measure).

We also compared the facility location function vs. the saturated coverage function. Figure 2.2 shows the performance of the facility location (f_{fac}) and saturated coverage (f_{sat}) functions in combination with the string kernel similarity measure. Comparing f_{fac} vs. f_{sat}, f_{sat} primarily controls for over-coverage of any element not in the chosen subset via the α saturation hyper-parameter. However, at a given α and for a given subset size, it does not guarantee that every non-selected element has good representation in the chosen subset. f_{sat} measures the quality of the subset by how well each individual element outside the subset has a surrogate within the chosen subset (via the max function), and hence can do better at

extreme size reductions (e.g., 2.5% - 10%).

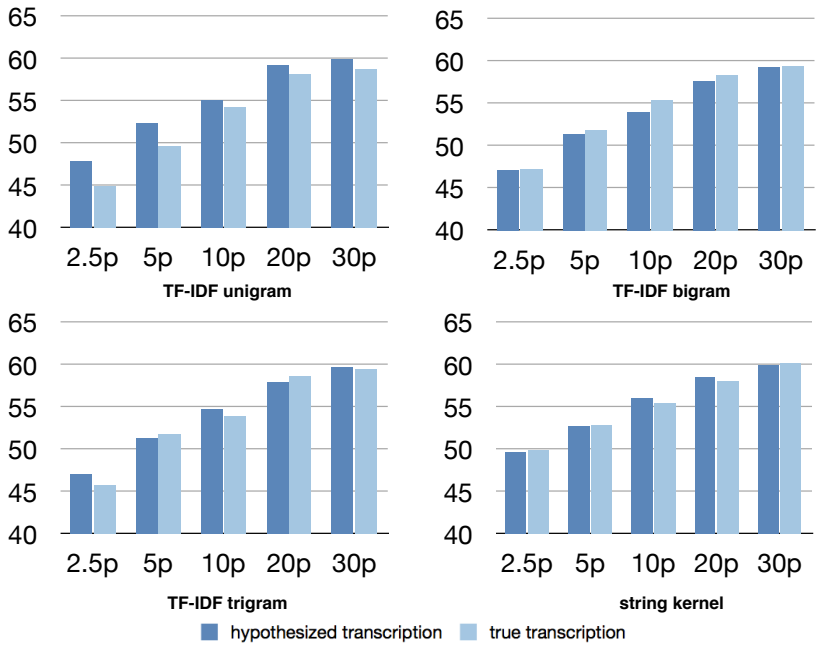


Figure 2.3: Phone accuracy for true vs. hypothesized phone labels, for string-based similarity measures.

Finally we examined whether using hypothesized phone sequences vs. the true transcriptions has negative effects. Figure 2.3 shows that this is not the case: interestingly, the hypothesized labels even result in slightly better results. This may be because the recognized phone sequences are a function of both the underlying phonetic sequences that were spoken and the acoustic signal characteristics, such as the speaker and channel. The true transcriptions, on the other hand, are able to provide information only about phonetic as opposed to acoustic characteristics.

Unsupervised Data Subset Selection

Next, we examine the performance of unsupervised submodular selection, which does not use the transcriptions. The different submodular objectives were f_{fac} and $f_{\text{2-fea}}$ (Equations 2.2 and 2.21, respectively). The similarity matrix in f_{fac} was computed by a gapped string kernel on tokenized utterances. In contrast to the supervised case, we utilized an HMM trained *in an unsupervised way* to produce the tokenization. This unsupervised model had 40 HMM states and 64 Gaussian mixture components per state and was trained on all of the training data. To instantiate the two-layer feature based function $f_{\text{2-fea}}$, we constructed the set of meta features U^2 as the set of tri-states extracted from the sequences of HMM state labels. Each tri-state $u_2 \in U^2$ was distinguished by the dominating Gaussian component index at the middle state; its corresponding lower-level features were constructed as the tri-state with all possible Gaussian component indices in the middle state. We constrained interactions between lower-level features and meta features to the case where both features shared the same tri-state, i.e., $W(u_1, u_2) = 1$ if u_1 and u_2 shared the same tri-state, and $W(u_1, u_2) = 0$ otherwise.

In addition to using an unsupervised HMM as the tokenizer we also tried using a k-component single-state unsupervised GMM. This model converted acoustic utterances into sequences of indices representing the dominant Gaussian component at each frame. A 512-component GMM was used to generate the set of low-level features U^1, and a 32-component GMM was used to generate meta features U^2. In both cases, we generate features as the Gaussian mixture indices for two consecutive frames (bigrams). The weight $W(u_1, u_2)$ was set to the co-occurrence count of features $u_1 \in U^1$ and $u_2 \in U^2$ in the training set. In both instantiations of $f_{\text{2-fea}}$, the concave functions $g_1()$ and $g_2()$ were set to the square root function, and the feature score function $m_u(A)$ was the sum of TF-IDF normalized feature counts.

Figure 2.4 shows the performance of the three unsupervised submodular selection methods described above. They all significantly ($p < 0.05$) outperform the random baseline for all

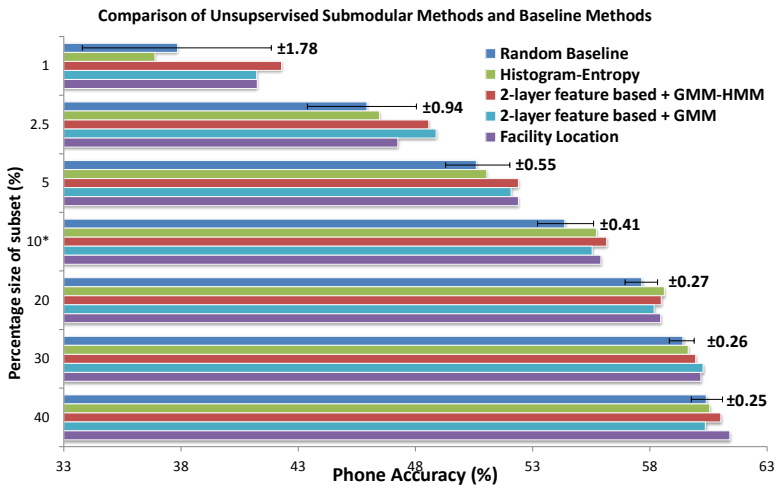


Figure 2.4: Unsupervised setting: Phone accuracy for different subset sizes; each block of bars lists, from top to bottom: random baseline, histogram-entropy baseline (supervised setting), $f_{\text{2-fea}}$+ GMM-HMM, $f_{\text{2-fea}}$+GMM, f_{fac}.

subset sizes except for $f_{\text{2-fea}}$+GMM at 40%. The improvement is more evident for small subset sizes (1%, 2.5%, 5%). In general, these unsupervised methods yield a performance comparable to that of the histogram-entropy baseline, which is a supervised method. In particular, $f_{\text{2-fea}}$+GMM-HMM outperforms the histogram-entropy baseline at all subset sizes, except for 20%.

2.2.2 LVCSR Experiments

Data and Systems

We evaluate our approach on selecting subsets from 1300 hours of conversational English telephone data from the Switchboard, Switchboard Cellular, and Fisher corpora. We train a separate ASR system on each resulting subset, where the sizes are chosen to be signif-

icantly smaller than the whole (namely, 1%, 5%, 10% and 20% of the whole non-silence training data). The development and test data sets are unchanged, and are the 2001 and 2002 NIST Rich Transcription development sets, with 2.2 hours and 6.3 hours of acoustic data, respectively. Two different ASR systems were used for our experiments, distinguished by their acoustic modeling approach. The first is SRI's DECIPHER system (Stolcke et al., 2000). The preprocessing component extracts 13-dimensional mel-frequency cepstral coefficients (MFCCs) along with their 1st, 2nd, and 3rd order derivatives. The resulting 52-dimensional feature vectors are mean and variance normalized, and reduced to 39 dimensions by heteroscedastic Linear Discriminant Analysis (HLDA) [Kumar and Andreou, 1997]. The features are then discriminatively transformed using feature minimum phone error training (fMPE) [Povey et al., 2005]. Acoustic models consist of three-state left-to-right HMMs with GMMs as output probability distributions. Each GMM represents a decision-tree clustered cross-word triphone state. GMMs are first estimated using the maximum likelihood criterion and are used to generate phone lattices, which are utilized for minimum phone error training (MPE) [Povey and Woodland, 2002] to create the final models. During decoding, a first-pass search is performed using a bigram language model. A recognition pass using maximum-likelihood linear regression (MLLR) speaker-adapted acoustic models generates a set of lattices. Finally, these lattices are rescored with a trigram language model to generate the final output.

The second system was also developed at SRI and utilizes DNNs as acoustic models and Kaldi [Povey et al., 2011] as the decoder. The inputs to the DNN consist of 15 frames of 40 dimensional Mel-scaled filter-bank outputs. The DNN targets are decision-tree clustered triphones with approximately 3750 targets for data subsets and 7800 for the full set.[3] The number of layers in each network was tuned based on the development set word error rate. As before, a first pass search is performed during decoding using a bigram language model,

[3]To clarify this point, the process of training a system using a subset (rather than all) of the training data was optimized in an attempt to get the best performance possible for any given amount of available data. Included in this process was the number of decision-tree clustered triphones for a given subset size.

and the resulting lattices are rescored using a trigram language model. The language model (LM) is the same in both systems and consists of an interpolation of various genre-specific LMs trained on meeting transcriptions, spontaneous telephone speech, broadcast news, and web data selected to match the transcribed data. The interpolation weights are optimized on a held-out set of meeting data.

We also compare our submodular data selection approach against two different baseline methods: random baseline, and the "histogram-entropy" baseline. For the random baseline we randomly sample data sets at the appropriate size (1%, 5%, 10%, or 20%), train different ASR systems for each set, and average their word error rates. For the histogram-entropy baseline, the objective is to select a subset of the data that results in a maximum-entropy distribution over linguistic units (e.g., triphone states or words) in the set. We implemented two variants of the baseline: (a) using the words from the true word transcriptions as phonetic units, as described in [Wu et al., 2007], and (b) using the triphone state labels from a forced alignment of the transcriptions to the acoustic data. For our experiments the fully-trained Decipher system was used for the forced alignment; note, however, that it is in principle also possible to use different acoustic models, or to perform an unsupervised tokenization of the acoustic signal. Also note that for the histogram-entropy method the objective criterion (maximum entropy of the distribution over phonetic units in the data) may saturate (i.e., no further increase is possible) before the budget constraint is reached. We found that this was the case for the 20% subset, i.e., the entropy saturates before 20% of the data has been selected. To reach 20%, more data would have to be added randomly, which would render the method largely equivalent to the random selection baseline. Results are unavailable for the histogram-entropy at 20% level.

Supervised Data Subset Selection

All experiments were conducted with the feature-based submodular function (Equation (2.4)). Several experiments were run with different instantiations of feature sets (words, triphones, triphone HMM state ids, and n-grams thereof), as well as different ways of normalizing

feature counts and different concave functions. The best results obtained with submodular selection were based on the square root function as the concave function g, the counts of triphones as the $m_u(s)$ scores, and the IDF weight of the triphone as the weight w_u for the corresponding feature u. The triphone annotation was also based on a forced-alignment of the word transcriptions to the signal. For each system and subset, the complexity of the acoustic model (number of initial clusters for bootstrapping the acoustic model, and the number of leaves in the decision tree used for state clustering) was optimized on the development set. Optimizing the number of parameters is important since data sets with greater inherent complexity can in theory support more parameters in the acoustic model, whereas inherently redundant data sets might lead to poorly trained models if too many parameters are utilized.

	1%	5%	10%	20%	all
Rand	52.1± 1.5	38.2±0.2	35.1±0.3	34.4±0.2	31.0
HE (words)	49.6	36.5	34.8	N/A	
HE (3-phones)	47.5	37.6	34.2	N/A	
SM (3-phones)	**47.5**	**35.7**	**33.3**	**32.6**	

Table 2.1: Word error rates for the HMM-GMM system, for subsets of various sizes chosen by the random (Rand), histogram-entropy (HE), and the submodular (SM) selection method. The histogram-entropy results for the 20% condition are not available due to that objective's saturation after 10%.

Table 6.1 shows the results for the HMM-GMM system. The random results are an average over four independent subsets for each size percentage, and are shown as mean ± standard_deviation. We see that our proposed method outperforms all baseline systems under all conditions.

Results for the DNN system are shown in Table 2.2. In this case, owing to the longer

	1%	5%	10%	20%	all
Rand	43.7±0.5	34.3±0.9	31.5±0.5	29.8±0.2	
HE (3-phones)	42.8	33.9	31.3	N/A	26.0
SM (3-phones)	**41.1**	**31.8**	**29.3**	**28.2**	

Table 2.2: Word error rates for the DNN system, for the random, histogram-entropy (HE) and the submodular (SM) training data subset selection methods.

training time, the random results are an average of three subsets for each size percentage. The histogram-entropy baseline and the submodular systems followed the same design as in the previous set of experiments. For all systems, the number of parameters (layers in the deep neural network) was optimized on the development set; it varied between three and six (with most systems having five layers). The number of hidden units was 1200 in each case. Here, too, the results shows that the submodular method clearly outperforms both baseline methods in all cases.

Comparing Table 6.1 to Table 2.2, moreover, shows an interesting trend. The submodular selected subset at 5% achieves a WER of 31.8% with the deep model system, while the HMM-GMM system using 100% of the training data achieves a WER of 31.0%. That is, the deep system using the "right" 5% of the training data almost matches the HMM-GMM system using all of the data. Note that the deep system is unable to do that, however, using the baseline methods for choosing 5% of the training data. Also, the deep system using only 10% of the submodular subselected data achieves a result (i.e., 29.3%) that is strictly better than the GMM system at 100% of the training data (i.e., 31.0%). These results, therefore, offer evidence of the combined power of a properly chosen subset of the training data (using a submodular function) and a modern speech recognition system (deep model based). Moreover, with such a large reduction (at 5%), the time spent training a deep system is approximately 20× faster, which could lead to many more model variants being

investigated in the same amount of time.

2.3 Discussion

In this chapter, we studied the speech data subset selection problem under both the supervised and unsupervised setting. The proposed submodular data summarization paradigm is a very general and efficient framework for such task. Mathematically, we showed that the data selection criteria proposed in [Siohan, 2014] and [Wu et al., 2007] can be equivalently modeled by our submodular framework with appropriate submodular functions. Moreover, our method has proved empirically effective on both small scale phone recognition and LVCSR tasks.

Future work will concentrate on selecting speech data for the active learning setting, where acoustic data is adaptively selected for labeling to further improve the system performance of an existing speech recognizer. Another direction of the future work is to select data in a less memory demanding scenario, such as the streaming setting.

Chapter 3

CASE STUDY II: GENOMICS ASSAY PANEL SELECTION

In our second case study, we apply the data summarization paradigm to the data collected from the biological domain. In particular, we focus on the genomics assay data, which is obtained by high-throughput DNA sequencing of the genomic positions. Genomics assays such as ChIP-seq, DNase-seq and RNA-seq can measure a wide variety of types of DNA activity. But their applications are significantly limited by their costs. In general, researchers would like to perform every type of genomics assay to characterize a cell type. Unfortunately, it costs about $400 [scienceexchange, 2015] to perform a single genomics assay with reasonable sequencing depth.

Even the ENCODE and Roadmap Epigenomics consortia [Bernstein et al., 2010, ENCODE Project Consortium, 2012], whose major responsibilities are to develop, perform, and analyze genomics assays, has only performed a total of about 1,359 assays as of January 2015. Among these assays, a total of 216 types of assays have been performed on at least one cell type, and at least one assay is done for a total of 228 cell types. On the other hand, it would be a total number of 49,248 assays if all types of assays are performed for all cell types. Despite the worldwide efforts with large budgets, the two consortia has only performed 5% of the possible number. Moreover, an infinite number of perturbations and variations of a given cell type exists. It would be of interest to study the various DNA activities on these cell types leading to an extremely large number of assays needed to be performed.

It is, therefore, critical to select a small panel of assays to perform on each cell type of interest—a problem we call *assay panel selection*—is a key step in any genomics project. So far, there has been little literature on how to choose such a panel. Most of the procedures for assay panel selection adopted in consortia such as ENCODE and Roadmap are based on

the investigators' domain knowledge and intuition.

In this Chapter, we give a more principled approach to address the problem based on the submodular data summarization paradigm. Our method aims to exploit the redundancy among assay types on the basis of existing data sets. In fact, many pairs of assay types yield similar information. Just to name a few, the transcription factors REST and RCOR1 are cofactors and therefore bind almost the same set of genomic positions [Andrés et al., 1999]. Similarly, the histone modification H3K36me3 primarily marks gene bodies, which are also transcribed and therefore measured by RNA-seq. Hence, a large fraction of the information learned from the full set of assay types may be extracted from only a small sized subset of representative assays.

As previously discussed, submodular functions are natural models for information and representativeness. Hence, a solution to the assay panel selection is to define an appropriate submodular function that characterizes the quality of a panel of assay types, and then to formulate the problem as a submodular optimization, for which one can efficiently find a panel with a high score according to the objective. Similar to what we have observed in the case study of speech data subset selection, we find that the facility location function is again a natural model and often yields superior performance on this task. Computing the facility location function requires a measure of similarity between assay types. For this purpose, we use the Pearson correlation between the two assays, averaged over the cell types in which the assays have been performed. As we will see, this objective function is chosen because it performs better than or comparably to the other methods. We call the resulting method *submodular selection of assays* (SSA).

Another important contribution of this case study is on the development of three evaluation mechanisms for automatically assessing the quality of any panel of assays. This contribution is independent of the proposed approach for the assay panel selection problem. In other words, the evaluation metrics will be useful for any future study of the quality of a panel of assays, independent of the particular procedure used to choose such a panel. The three evaluation metrics, each of which assesses the quality of a panel with respect to a

distinct application, are described as follows:

1. Assay imputation: the accuracy with which the panel can be used to impute the results of assays not included in the panel;

2. Functional element prediction: the accuracy with which the panel can be used to detect functional elements such as transcription factor binding sites, promoters, and enhancers;

3. Annotation-based evaluation: the quality of a whole-genome annotation produced using the panel.

In addition to the evaluations metrics, we investigate the assay panel selection problem under two settings. We call the first setting – the "future" variant. It arises when a researcher is interested in applying a panel of genomics assays to a new tissue type or cellular condition. In this case, the researcher must use previously performed assays in other cell types to choose a representative panel of assay types. The second setting we consider is called the "past" variant. It arises, on the other hand, when one is interested in applying a computationally expensive analysis, such as a genome annotation method, that cannot efficiently be run on all available data sets. In this setting, the researcher must choose a representative panel from the available data to use as input to the analysis. In this case, the researcher may use the data from assays performed on the cell type in question to inform their choice. We propose a variant of our method, called SSA-past, which leverages this information to allow a researcher to choose such a representative panel in this setting. In this work, we are primarily interested in the "future" setting.

We illustrate the assay panel selection problem along with the selection process by SSA schematically in Figure 3.1. In summary, the method takes as input all available existing genomics assays, where each assay is represented as a real-valued track over the genome. In the SSA-past mode, SSA selects a panel of already-performed assays to use as input to an expensive computational analysis. In the SSA-future mode, SSA chooses a panel of assay

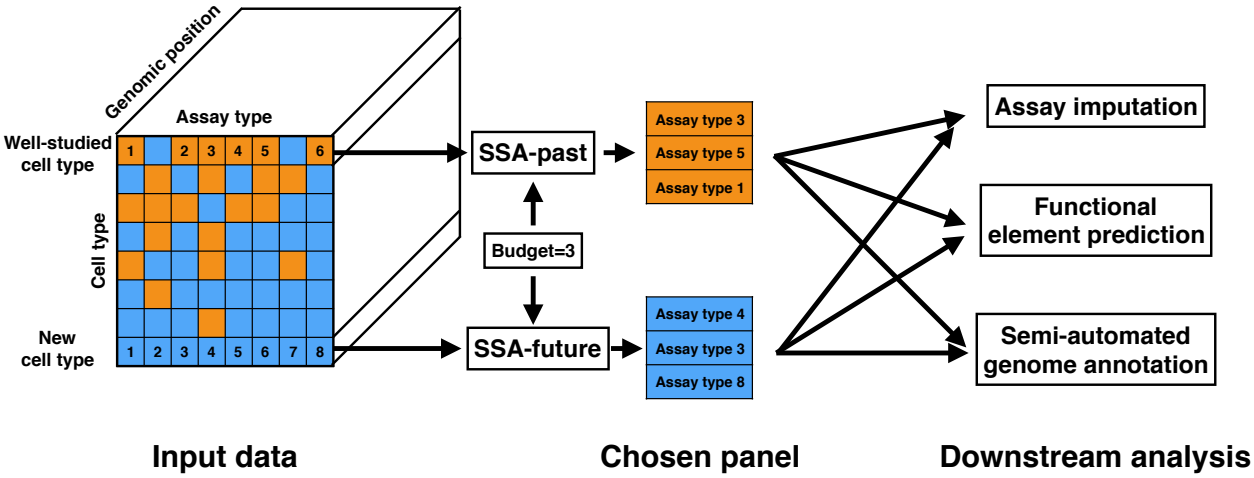


Figure 3.1: Schematic illustration of the assay panel selection problem, selection process, and evaluation mechanisms.

types to be performed in a new cell type. In both cases, the resulting data sets are provided as input to downstream analyses, which may include imputing assays that weren't performed, predicting the locations of functional elements, or semi-automated genome annotation.

3.1 Framework for Submodular Selection of Assays

In this section, we describe the proposed framework – SSA in detail. We first define the useful notations that facilitates the description of the method. We then define the problem of assay panel selection as a constrained submodular optimization, where the facility location function is the objective. We lastly describe the technique for computing the similarity between assay types, which is useful for instantiating the facility location function.

3.1.1 Notation

We use the following notation below to facilitate the description of our method. We use the term "assay type" to mean a particular genomics assay protocol that may be performed in any cell type (for example "ChIP-seq targeting H3K27me3") and "assay" to mean a particular assay type performed in a particular cell type. The term "cell type" refers to any cellular state that may be queried with a genomics assay, which may refer to any combination of cell line, tissue type, disease state (such as cancer), individual, or drug perturbation. We refer to a cell type as c and the entire set of all cell types as $\mathcal{C}$ ($|\mathcal{C}| = 228$). We use a to refer to an assay type, A for a subset of assay types, and $\mathcal{A}$ for the set of all assay types ($|\mathcal{A}| = 216$). We use s to denote a single assay (that is, a given assay type performed in a given cell type), S for a set of assays, and $\mathcal{S}$ as the set of all available performed assays. Given any cell type $c \in \mathcal{C}$ we define the set of assay types performed in this cell type as $\mathcal{A}^c$ and the corresponding assays as $\mathcal{S}^c$. We define $I = \{1, \ldots, n\}$ as the set of all positions in a genome. An assay s is represented as a vector of length n; i.e., $s \in \mathbb{R}^n$. We denote its ith entry (i.e., the value of assay s at genomic position i) as $s(i)$.

3.1.2 Problem Formulation

Similar to the speech data subset selection framework, we are given a set $V = \{v_1, \ldots, v_m\}$ of assay types and a monotone submodular function $f : 2^V \rightarrow \mathbb{R}$ that valuates the quality of any subset $A \subseteq V$ as $f(A)$. Given the constraint of selecting no more than k assay types, the assay panel selection problem can be formulated as follows:

$$\text{maximize } f(S), \quad \text{subject to } |S| \leq k \tag{3.1}$$

for some integer $k \leq |V|$. It is in the form as described in Problem 2, which has been extensively studied in the submodular optimization literature. The simple greedy algorithm (Alg. 3) approximately solves the problem with a constant factor $(1 - 1/e)$.

In this work we use the *facility location* function to measure the quality of panel of assay types. Recall that the facility location function $f_{\text{fac}} : 2^V \to \mathbb{R}$ has the following definition:

$$f_{\text{fac}}(S) = \sum_{s' \in V} \max_{s \in S} r_{s',s}, \tag{3.2}$$

where $r_{s',s}$ measures the pairwise similarity between assays s' and s (defined below). Intuitively, the facility-location function takes a high value when every assay in V has at least one similar representative in S.

We define this similarity between assay types differently depending on the application: In the selection of past assays setting, the particular assays performed in the cell type of interest c are available, while in the selection of future assays setting we must estimate this similarity from reference cell types. In the selection of past assays setting, we directly use the signal vectors s_i and s_j to derive the similarity. We define this similarity as $r_{s_i,s_j} = |\rho_{s_i,s_j}| \in [0, 1]$, where ρ_{s_i,s_j} is the Pearson correlation between the signal vector s_i and s_j. Pearson correlation is frequently used to evaluate the similarity between genomics assays [ENCODE Project Consortium, 2012]. For efficiency, we compute the correlation measure ρ_{s_i,s_j} only across a subset of genomic positions $I' \subseteq I$, where I' is randomly subsampled from I, and $|I'| \approx 0.01|I|$.

In the selection of future assays setting, the assays in the cell type c are not available, but the assays performed in cell types other than c, $\mathcal{S} \setminus \mathcal{S}^c$, are available. Let $a_i, a_j \in \mathcal{A}$ be the assay types associated with the assay s_i and s_j, respectively. Let $\mathcal{S}^{a_i}$ be the set of assays in $\mathcal{S}$ with type a_i. We approximate the similarity between s_i and s_j by aggregating the all similarity between the pairs in $\mathcal{S}^{a_i} \setminus s_i$ and $\mathcal{S}^{a_j} \setminus s_j$. We utilize the aggregation strategy by taking the average of these similarity scores. Mathematically, the aggregated similarity is defined below:

$$r_{s_i,s_j} \triangleq \frac{1}{|\mathcal{S}^{a_i} \setminus s_i|} \frac{1}{|\mathcal{S}^{a_j} \setminus s_j|} \sum_{s \in \mathcal{S}^{a_i} \setminus s_i} \cdot \sum_{s' \in \mathcal{S}^{a_j} \setminus s_j} |\rho_{s,s'}|, \tag{3.3}$$

We chose to use the average correlation r because the facility location function defined via the similarities aggregated in this way correlated best with our evaluation metrics. We provide

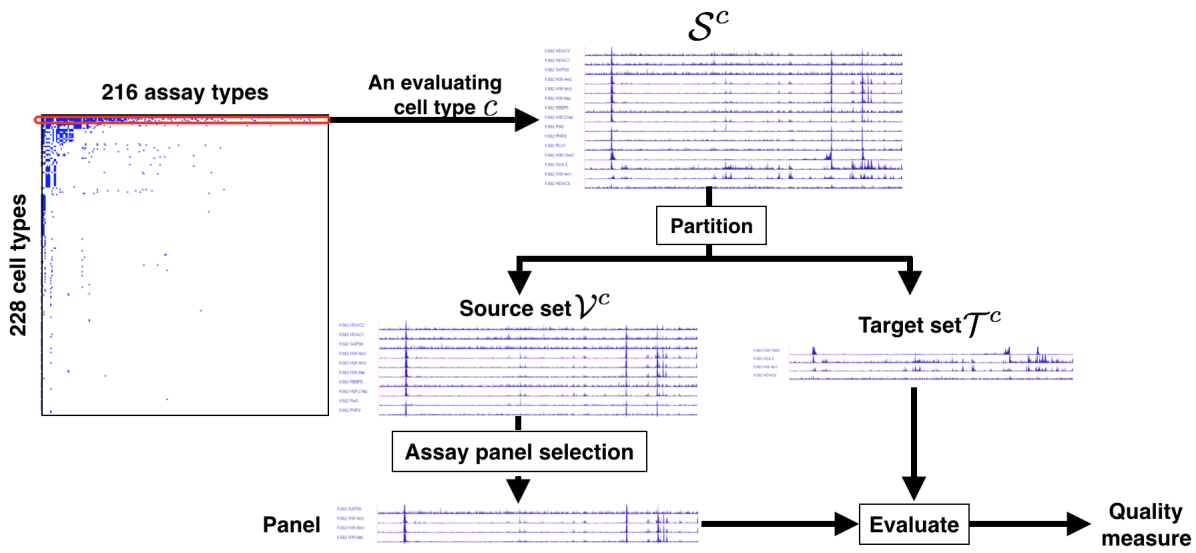


Figure 3.2: The overall cross-validation evaluation strategy.

the comparison of such aggregation strategy against other strategies in the experimental section.

3.2 Evaluation Methods

In this section, we focus on the method for evaluating the quality of a panel of assay types. We first define a useful cross-validation strategy and then introduces three classes of evaluation metrics that characterize the quality of a panel for distinct applications.

3.2.1 Cross-validation strategy

We would prefer to apply our method once to select a single panel of assay types. However, doing so could result in a panel of assay types that have not been performed in any cell type (or very few cell types), which would prohibit evaluating the quality of this panel. Therefore, we apply a cross-validation strategy that repeatedly holds out a subset of assay types for evaluation and selects a panel from the remaining assay types, and we perform this cross-

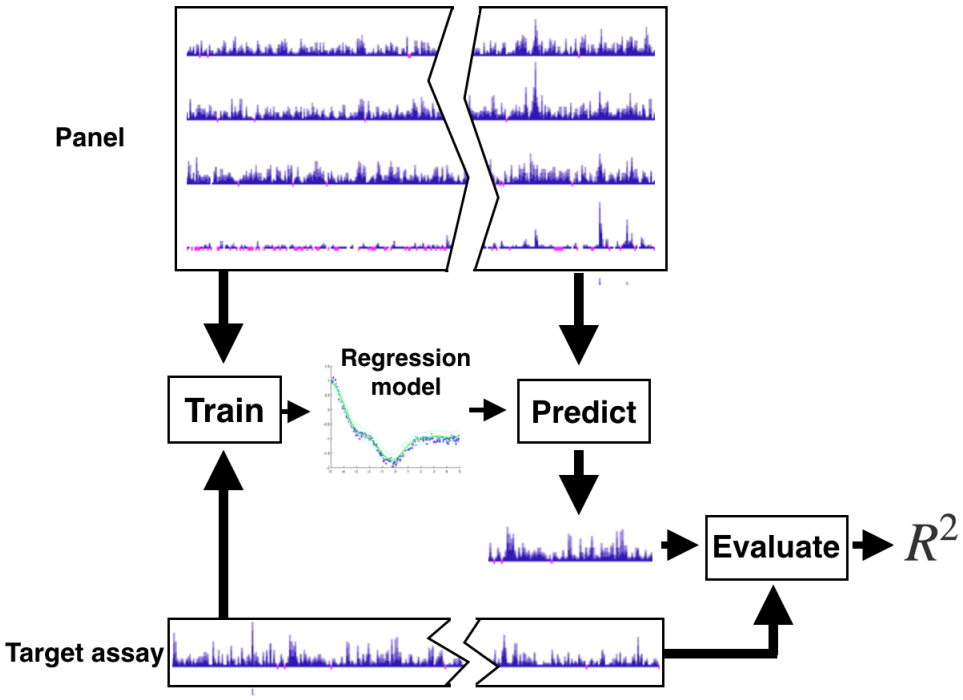


Figure 3.3: Schematic of the assay imputation.

validation separately for each cell type in turn (Figure 3.2). To evaluate the quality of our method with respect to a cell type c, we restrict ourselves to selecting from the set of assays performed in c ($\mathcal{S}^c$). We randomly partition $\mathcal{S}^c$ into 10 equally-sized, disjoint folds. Of the 10 folds, a single fold is retained as the target set $\mathcal{T}^c$, and the remaining 9 blocks are used as the source set $\mathcal{V}^c$. We select a panel of assays $S \subseteq \mathcal{V}^c$ from the source set $\mathcal{V}^c$ and evaluate the panel on the assays relative to the target set $\mathcal{T}^c$ using the three evaluation metrics described below. The process is then repeated ten times, with each of the ten folds used once as the target set. We average the ten results are averaged to produce a single number representing the performance.

3.2.2 Assay imputation

The *assay imputation* evaluation metric measures the ability of a panel of assay types to be used to impute the results of other assay types outside the panel (Figure 3.3). We formalize

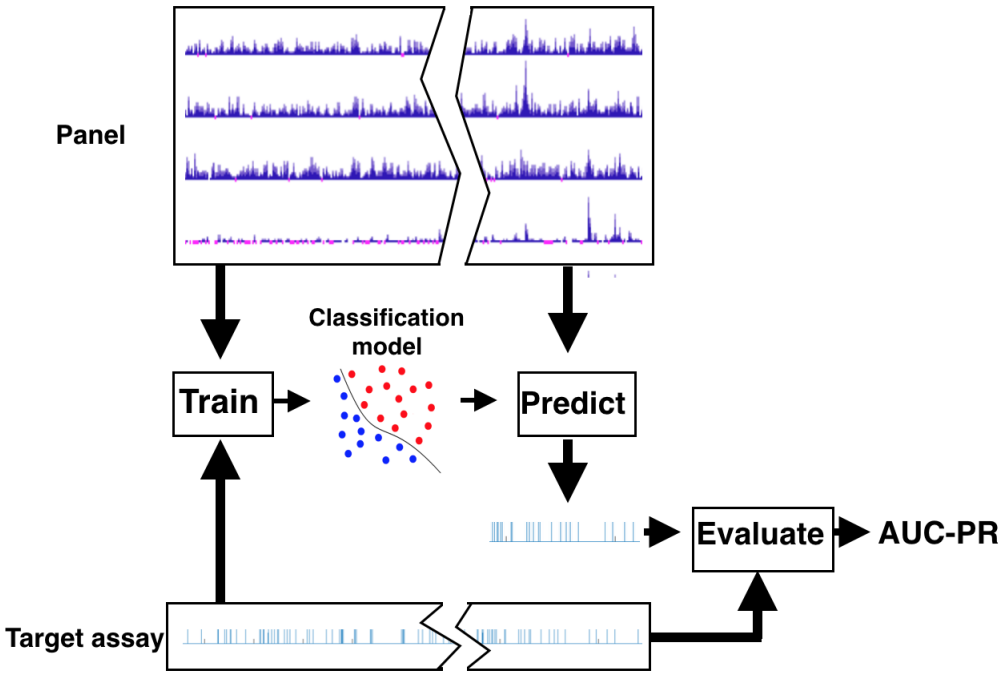


Figure 3.4: Schematic of the functional element prediction.

assay imputation metric as a regression problem in which the assays in the panel S are used as features to predict the target set assays, $s' \in \mathcal{T}^c$. In this regression problem we have one labeled example for each position in the genome.

As our regression model, we use support vector regression with a Gaussian kernel. To construct the training and test data, we randomly choose disjoint sets of genomic positions $I^{Tr}, I^{Te} \subseteq I$, where $I^{Tr} \cap I^{Te} = \emptyset$. In our experiments, we set $|I^{Tr}| = 5,000$ and $|I^{Te}| = 2,000$. Given the panel $S = \{s_1, \ldots, s_{|S|}\}$, a target assay $s' \in \mathcal{T}^c$, and the training genomic positions I^{Tr}, we create the training data as $\mathcal{D}^{Tr} = \{x^i, y^i\}_{i \in I^{Tr}}$, where $x^i = [s_1(i), s_2(i), \ldots, s_{|S|}(i)]^T$ and $y^i = s'(i)$. Similarly, the test data set is constructed as $\mathcal{D}^{Te} = \{x^i, y^i\}_{i \in I^{Te}}$. The hyperparameters of the regression model are tuned using 5-fold cross validation. We measure the performance of the trained model on the test data $\mathcal{D}^{Te}$ as the squared correlation coefficient $\theta_{s'}$. We repeat this evaluation process for every target assay in $\mathcal{T}^c$ and report the performance of the panel S as the average squared correlation coefficient $\theta = \frac{1}{|\mathcal{T}^c|} \sum_{s' \in \mathcal{T}^c} \theta_{s'}$.

3.2.3 Functional element prediction

The *functional element prediction* evaluation metric evaluates how well a panel of assays can predict the genomic locations of functional elements such as promoters, enhancers and insulators. Because there are few validated examples of each type of element, we use experimentally-determined binding of transcription factors, as determined by transcription factor ChIP-seq peaks, as a proxy for functional elements. Most known types of functional elements can be characterized by the binding of particular transcription factors [Visel et al., 2009, Burgess-Beusse et al., 2002]. Note that functional element prediction is similar to assay imputation in the sense that both evaluation metrics aim to predict the output of a genomics assay; however, functional element prediction focuses on just transcription factor binding sites, whereas assay imputation focuses on the whole genome. Similar to assay imputation, we consider this metric separately for each cell type. For an evaluation cell type c, we denote the set of transcription factor ChIP-seq assays performed in c as $\hat{\mathcal{S}}^c \subseteq \mathcal{S}^c$. Given a bi-partition of $\mathcal{S}^c$ into the source set $\mathcal{V}^c$ and the target set $\mathcal{T}^c$, we choose from the source set $\mathcal{V}^c$ a panel of assays, and we evaluate functional element prediction only on the target assays in the set $\hat{\mathcal{T}}^c = \mathcal{T}^c \cap \hat{\mathcal{S}}^c$, in contrast to the assay imputation metric where all assays in $\mathcal{T}^c$ are used for evaluation.

For a target transcription factor assay $s \in \hat{\mathcal{T}}^c$, let p be a binary vector $\{0,1\}^n$ indicating the genomic positions where s has a peak as called by the peak-calling algorithm. That is, $p(i) = 1$ if there is a peak at position i, and $p(i) = 0$ otherwise. We use a support vector machine (SVM) with Gaussian kernel to predict p given a panel of assays $S \subseteq \mathcal{V}^c$. For a given testing factor p, we refer to the positions where $p = 1$ as I_+ and the set of positions where $p = 0$ as $I_- = I \setminus I_+$. We randomly choose $I_+^{Tr} \subseteq I_+$ and $I_-^{Tr} \subseteq I_-$ as the positive and negative positions to generate training samples. Similarly, the testing samples are randomly chosen from $I_+^{Te} \subseteq I_+ \setminus I_+^{Tr}$ and $I_-^{Te} \subseteq I_- \setminus I_-^{Tr}$. Given the panel $S = \{s_1, \ldots, s_{|S|}\}$ of assays and the set of positive training genomic positions I_+^{Tr}, we construct the set of positive training samples as $\mathcal{D}_+^{Tr} = \{x^i, +1\}_{i \in I_+^{Tr}}$ where $x^i = [s_1(i), \ldots, s_{|S|}(i)]^T$. Similarly, we construct the

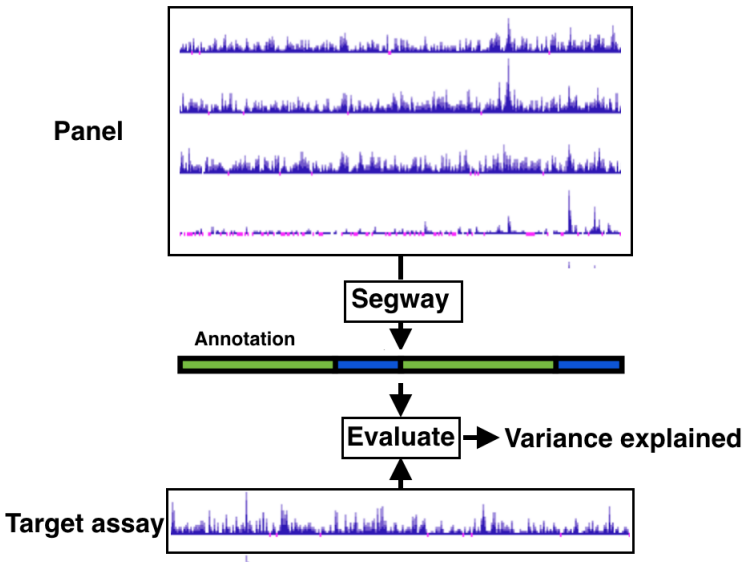


Figure 3.5: Schematic of the annotation-based evaluation metric.

negative training samples, positive test samples, and negative test samples as $\mathcal{D}_{-}^{Tr}$, $\mathcal{D}_{+}^{Te}$, and $\mathcal{D}_{-}^{Te}$, respectively. The SVM is first trained on the training data set $\mathcal{D}^{Tr} = \{\mathcal{D}_{+}^{Tr}, \mathcal{D}_{-}^{Tr}\}$, and then evaluated on the testing data set $\mathcal{D}^{Te} = \{\mathcal{D}_{+}^{Te}, \mathcal{D}_{-}^{Te}\}$.

Because there are far more genomic positions that are not a functional element than there are positions that are, measures of predictive accuracy such as the total fraction of correct predictions ("accuracy") and the area under the receiver operating characteristic curve do not offer a reasonable measure of performance. Instead, we compute the area under the curve of a precision-recall plot (AUC-PR), which is particularly well suited for settings with imbalanced class distributions [Craven and Bockhorst, 2005, Davis and Goadrich, 2006]. In our experiments we set $|I_{+}^{Tr}| = 200$, $|I_{-}^{Tr}| = 20,000$, $|I_{+}^{Te}| = 100$ and $|I_{-}^{Te}| = 10,000$. We apply 5-fold cross validation for tuning the hyperparameters of the SVM. Let $\gamma_{s'}$ be the normalized area under curve for the precision-recall plot (i.e., $\gamma_{s'} \in [0,1]$) for each target assay $s' \in \hat{\mathcal{T}}^c$. We illustrate this procedure schematically in Figure 3.4. We report the performance as the average AUC-PR on all target assays, i.e., $\gamma = \frac{1}{\hat{\mathcal{T}}^c} \sum_{s' \in \hat{\mathcal{T}}^c} \gamma_{s'}$.

3.2.4 Annotation-based evaluation

The *annotation-based evaluation* metric measures the quality of a panel of genomics assays according to the quality of the genome annotation that is obtained by inputting the panel into a semi-automated genome annotation (SAGA) algorithm. SAGA algorithms are widely used to jointly model diverse genomics data sets. These algorithms take as input a panel of genomics assays and simultaneously partition the genome and label each segment with an integer such that positions with the same label have similar patterns of activity. These algorithms are considered "semi-automated" because a human performs a functional interpretation of the labels after the annotation process. Examples of SAGA methods include HMMSeg [Day et al., 2007], ChromHMM [Ernst and Kellis, 2010], Segway [Hoffman et al., 2012] and others [Thurman et al., 2007, Lian et al., 2008, Filion et al., 2010]. These genome annotation algorithms have had great success in interpreting genomics data and have been shown to recapitulate known functional elements including genes, promoters and enhancers. We use the SAGA method Segway in this work.

In order to apply annotation-based evaluation to a panel of assays, we input this panel into a SAGA algorithm and evaluate the resulting annotation (Figure 3.5). Intuitively, a diverse panel of assays input to a SAGA algorithm should more accurately capture important biological phenomena than a redundant panel. To evaluate the quality of an annotation relative to a particular genomics data set, we use the *variance explained* measure [Libbrecht et al., 2015]. Given an evaluation cell type c we randomly partition $\mathcal{S}^c$ into a source set $\mathcal{V}^c$ and a target set $\mathcal{T}^c$. For a given panel of assays $S \subseteq \mathcal{V}^c$, we first train a Segway model based on the panel and then obtain an annotation y. Segway outputs an annotation $y \in \mathcal{Y}^n$, where $\mathcal{Y} = \{1, 2, \ldots, k\}$ is a set of k labels that an annotation can take on at each genomic position. For each target assay $s' \in \mathcal{T}^c$, we measure the quality of the annotation y as how well it explains the variance of the assay s'. We first compute the signal mean of s' over the positions assigned a given label ℓ as

$$\mu_\ell \triangleq \frac{\sum_{i=1}^{n} \mathbf{1}(y(i) = \ell) s'(i)}{\sum_{i=1}^{n} \mathbf{1}(y(i) = \ell)} \quad \text{for } \ell \in \{1 \ldots k\}. \tag{3.4}$$

We then define a predicted signal vector $\hat{s}'$ with $\hat{s}'(i) = \mu_{y(i)}$ and compute the prediction error as $d_i = \hat{s}'(i) - s'(i)$. We compute the residual standard deviation of the signal vector as

$$\sigma_{\text{res}} \triangleq \text{stdev}(d_{1:n}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \text{mean}(d_{1:n}))^2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_i^2}. \tag{3.5}$$

The last equality holds because $\text{mean}(d_{1:n}) = 0$ by construction. σ_{res} measures the residual standard deviation of the target assay s' accounting for the annotation y. Let $\sigma_{\text{OV}} = \text{stdev}(s'(1:n))$ be the overall standard deviation of the assay s'. The normalized variance explained by the annotation y is then

$$\alpha_{s'} = \frac{\sigma_{\text{ov}} - \sigma_{\text{res}}}{\sigma_{\text{ov}}}. \tag{3.6}$$

Observe that σ_{ov} always upper bounds σ_{res}. The measure $\alpha_{s'} \in [0, 1]$ represents the fraction of the variance of the assay s' explained by the annotation y, where larger values indicate better agreement.

In our experiments, we trained the Segway model with 10 EM random initializations and 15 labels at 100 base-pair resolution. We report the performance as the averaged measure on all target assays as $\alpha = \frac{1}{|\mathcal{T}^c|}\sum_{s' \in \mathcal{T}^c} \alpha_{s'}$.

3.3 Experimental Results

3.3.1 Genomics Data

We acquired all public genomics data from the ENCODE (`http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/`) and Roadmap Epigenomics (`https://sites.google.com/site/anshulkundaje/projects/epigenomeroadmap`) projects as of January 2015. These data sets were processed by the two consortia into real-valued data tracks, as described previously [Hoffman et al., 2013, Kundaje et al., 2015]. We omitted all assays with more than 1% unspecified positions, which may indicate errors during processing or mapping. We manually curated these assays to unify assay type and cell type terminology and, when multiple assays were available, we arbitrarily chose a representative assay for each (cell type, assay

type) pair. This procedure resulted in a total of 1,359 assays comprised of a total of 216 assay types and 228 cell types. The assay types include ChIP-seq with a variety of targets (both histone modification and transcription factor), DNase-seq, FAIRE-seq, Repli-seq and RNA-seq. We applied the inverse hyperbolic sine transform $\text{asinh}(x) = \ln(x + \sqrt{x^2 + 1})$ to all signal data. This function has the compressing effect of a function like $\log x$ for large values of x but it is defined at zero and has much less of a compressing effect for small values. The asinh transform has been shown to be important for reducing the effect of large values in analysis of genomics data sets [Johnson, 1949, Hoffman et al., 2012]. Transcription factor ChIP-seq peaks were called by each consortium for each factor using MACS using an irreproducible discovery rate (IDR) threshold of 0.05 [Zhang et al., 2008, Landt et al., 2012].

3.3.2 Submodular selection of assays identifies diverse panels of genomics assays

Because researchers generally perform panels of either histone modifications or transcription factor ChIP-seq assay types but rarely perform mixed panels, we run the method separately on transcription factor and histone modification types. We first experiment with the "future" setting. We implement SSA with the facility location function, whose similarity measure between assay types is derived according to Eqn. 3.3.

By analyzing data from the ENCODE and Roadmap Epigenomics Consortia, we found that SSA results in assay panels with diverse genomic functions. When choosing from transcription factors, SSA chooses factors that engage in diverse regulatory pathways (Table 3.1). The vast majority of transcription factors in our data set bind to promoters and enhancers and regulate the transcription of RNA Pol II-transcribed genes. The top five transcription factors chosen by SSA include three of these factors, each of which regulate very different regulatory pathways: SMARCB1, an ATP-dependent chromatin remodeler; PML, a tumor suppression factor; and STAT5A, a factor involved in developmental signal transduction [Consortium, 2014]. The top five also includes two factors—CTCF and BRF2—that are not solely involved in RNA Pol II-mediated transcription. CTCF, part of the cohesin complex, regulates chromatin conformation and enhancer-promoter insulation, and only about

Choice order	Transcription factor	Function	Singleton score	Objective gain
1	SMARCB1	ATP-dependent chromatin remodeling	14.78	14.78
2	PML	Tumor suppression	14.12	1.84
3	STAT5A	Developmental signal transduction	14.17	0.91
4	CTCF	Chromatin conformation and insulation	8.40	0.81
5	BRF2	RNA polymerase III initiation complex	8.88	0.55

Table 3.1: Panels of transcription factors assays chosen by SSA-future. Each assay type is in the order assigned by SSA; for any size k, the top k assay types in the list are the chosen panel of this size. The "singleton score" is the objective value of a panel containing only the indicated assay type, and the "objective gain" indicates the improvement in objective that results from SSA adding the indicated assay type to the growing panel. Because there are 80 transcription factors, we display just the top five chosen by SSA. Associations are summarized from UniProt [Consortium, 2014].

half of its binding sites occur in promoters or enhancers. BRF2 is part of the RNA Polymerase III complex, which transcribes rRNA, tRNA and other small RNAs. These two assay types each have low objective scores when in a panel by themselves ("singleton scores"), but are chosen by SSA because they measure different types of activity than the rest of the panel. Therefore, they are important to include in a diverse panel.

When choosing a panel of histone modifications, SSA selects marks that cover diverse types of genomic regions and exhibit qualitatively different patterns (Figure 3.6 and Table 3.2). The top six histone modifications include a promoter mark (H3K4me3), an enhancer mark (H3K4me1), a gene mark (H3K79me2) and marks associated with both known types of repressive domains, facultative (H3K27me3) and constitutive (H3K9me3) heterochromatin. The two repressive marks, H3K27me3 and H3K9me3, have the lowest singleton scores of all

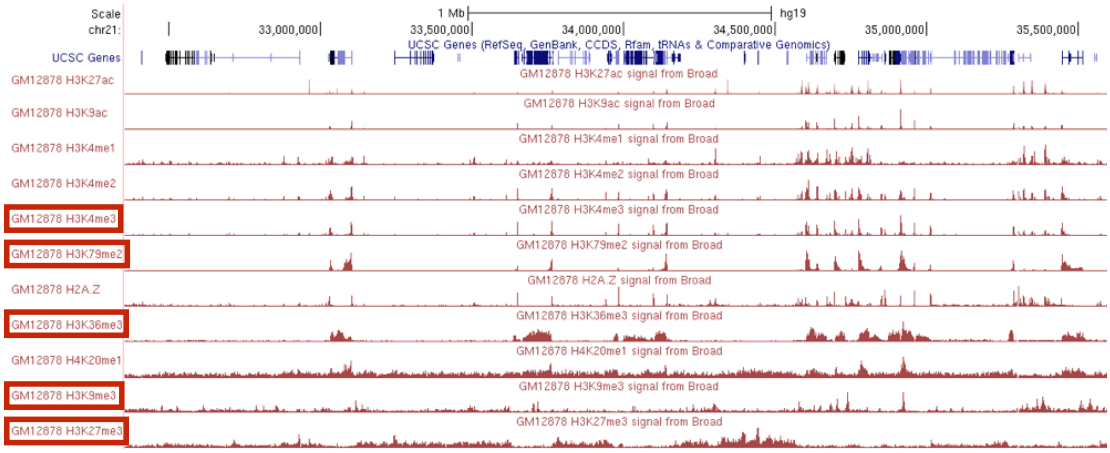


Figure 3.6: Redundancy in histone modification signal in the genome. The top five assay types chosen by SSA are boxed in red.

the histone modifications, but they give a high objective gain because they measure distinct activity from the rest of the panel. In contrast, even though H3K27ac is sometimes considered the best individual mark of enhancers and has a high singleton score, it is chosen last by SSA because it is redundant with other assay types in the panel, such as H3K4me1 and H3K9ac. The top six includes two different marks of transcription, H3K79me2 and H3K36me3, but these two modifications mark different parts of genes and are regulated differently relative to the gene's level of transcription [Li et al., 2007]. As expected, SSA ranks additional measures of regulation (H3K4me2, H2A.Z, H3K9ac and H3K27ac) low on the list because these marks are redundant with the regulatory marks H3K4me1 and H3K4me3.

SSA almost exactly recapitulates the panel of histone modifications chosen by the Roadmap Epigenomics consortium (boldface entries in Table 3.2). This consortium chose a set of five "core" histone modifications to assay across 111 human primary tissues. This choice was made by the members of this consortium based on their collective, expert knowledge. These five core histone modifications ranked among the top six modifications chosen by SSA. In fact, the SSA-chosen and Roadmap-chosen panels of size 5 have very similar scores according to the facility location function, ranking 1 and 16 respectively out of all $\binom{11}{5} = 2772$ possi-

Choice order	Histone modification	Association	Singleton score	Objective gain	Objective loss if swapped in
1	**H3K4me3**	Promoters	3.18	3.18	
2	H3K79me2	Transcription	2.40	0.97	
3	**H3K9me3**	Constitutive heterochromatin	0.70	0.34	
4	**H3K27me3**	Facultative heterochromatin	0.86	0.25	
5	**H3K36me3**	Transcription	1.21	0.23	
6	**H3K4me1**	Enhancers	1.86	0.18	0.05 (H3K36me3)
7	H3K4me2	Regulatory	3.12	0.08	0.07 (H3K36me3)
8	H3K9ac	Regulatory	3.13	0.07	0.15 (H3K36me3)
9	H2A.Z	Promoters	2.47	0.05	0.16 (H3K27me3)
10	H4K20me1	Transcription	1.41	0.005	0.22 (H3K36me3)
11	H3K27ac	Regulatory	2.61	0	0.14 (H3K36me3)

Table 3.2: Panels of histone modification assays chosen by SSA-future. See the text for a description of the "Objective loss if swapped in" column. There are only eleven histone modifications, so we display the full list. Bold font indicates those histone modification assays chosen by the Roadmap Epigenomics consortium.

ble panels of five histone modifications. Therefore, SSA closely reproduces careful, manual selection by experts in an entirely automated and data-driven way.

To better understand the choices made by SSA in selection of histone modifications, we performed a "swap-out" experiment (final column of Table 3.2). We started with the panel of size five selected via SSA, and we asked, for each of the remaining six histone modification assays, how much the objective function would decrease if we were forced to swap one of the SSA-selected assays for the excluded assay. In five out of the six cases, the objective is maximized by swapping the excluded assay with the last-selected histone modification, H3K36me3. However, the magnitude in the change in objective varies quite a bit: swapping in H3K4me1 makes very little difference (0.05), whereas swapping in H3K36me3 yields a relatively large change in objective (0.22). This type of exploratory analysis can be quite

valuable in the context of a real experimental design setting, where qualitative features of the assays (e.g., familiarity to the researchers involved) are important but difficult to quantify.

3.3.3 Choose the best variant of SSA

In this section, we empirically evaluate a number of variants of SSA via the panel evaluation framework as proposed in Section 3.2. We compare among various submodular functions as the objective for SSA. We also examine various aggregation strategies for similarity computation. A comparative study is performed on different supports of the genomics regions for deriving assay type similarities. We also analyze the difference in performance by using Pearson correlation versus Spearman correlation for similarity computation.

First, to determine the most effective objective function, we compared the facility location function and four other potential objective functions based on the pairwise similarity matrix. The potential objectives include the saturated coverage function f_{sat}, diversity function f_{div}, and log determinant function $f_{\text{log-det}}$. The definitions of these functions are given below:

1. *Saturated coverage function:*

$$f_{\text{sat}}(A) = \sum_{v \in V} \min\{\sum_{a \in A} r_{v,a}, \beta \sum_{v' \in V} r_{v',v}\}, \tag{3.7}$$

   where $0 \leq \beta \leq 1$ is a hyperparameter that controls the saturation of the coverage for each item $v \in V$. In the experiment we set $\beta = k/n$, where k is the target number of assays to select, and n is the size of entire set V of all available assays.

2. *Diversity function:*

$$f_{\text{div}}(A) = -\sum_{a \in A} \sum_{a' \in A} r_{a,a'} \tag{3.8}$$

3. *Log determinant function:*

$$f_{\text{log-det}}(A) = \log\det(I + \lambda S_A), \tag{3.9}$$

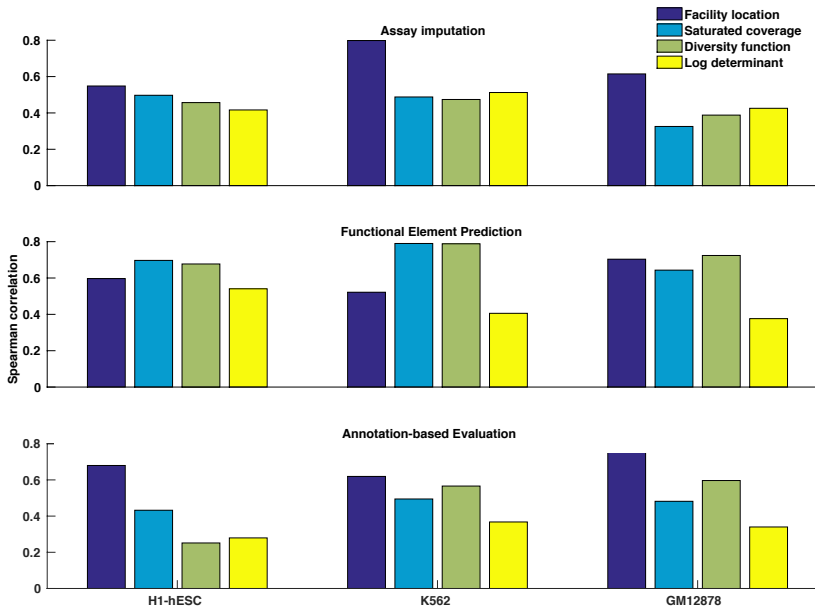


Figure 3.7: Comparison among various submodular functions in terms of the Spearman correlation between function valuation and the performance metrics.

where $\lambda > 0$ is a hyperparameter, and S_A is the pairwise similarity matrix indexed by the subset $A \subseteq V$.

We measure the performance of these submodular objectives by examining how their objective valuations correlate with the three proposed evaluation metrics (assay imputation, functional element prediction, and annotation-based evaluation). The experiment is performed under the selection of past assays setting, where we compute the similarity between a pair of assays using the Pearson correlation. Given an evaluation metric, a cell type c, a selection budget K, and a submodular objective f, we randomly draw n subsets $\{A_i\}_{i=1}^n$ of assays from the cell type c, where each subset A_i is of size k. We then compute the Spearman correlation between the submodular valuation $\{f(A_i)\}_{i=1}^n$ and the performance measure $\{y_i\}_{i=1}^n$ under the evaluation metric. We report the correlation measure averaged over budget constraints $k = 3, 4, 5, 6$. In the experiment we set $n = 20$, and we test on the cell types

K562, H1-hESC, and GM12878. The results (Figure 3.7) suggest that the facility location function, in most cases, yields the highest correlation with the three evaluation metrics.

Next, we study how the performance of SSA for the selection of future assays scenario varies with different choices of similarity aggregation strategies. We focus on strategies for defining the pairwise similarity between assay types in an evaluating cell type c given the assays performed on cell types other than c. We consider the six aggregation strategies r^1, r^2, r^3, r^4, r^5, and r^6, where they take the average, 0^{th}, 25^{th}, 50^{th}, 75^{th}, and 100^{th} percentile over the available similarity scores, respectively. For completeness, we also give their corresponding mathematical definition below:

$$r_{s_i,s_j} \triangleq \frac{1}{|\mathcal{S}^{a_i} \setminus s_i|} \frac{1}{|\mathcal{S}^{a_j} \setminus s_j|} \sum_{s \in \mathcal{S}^{a_i} \setminus s_i} \cdot \sum_{s' \in \mathcal{S}^{a_j} \setminus s_j} |\rho_{s,s'}|, \tag{3.10}$$

$$r^2_{s_i,s_j} \triangleq \text{percentile}(\{|\rho_{s,s'}| : s \in \mathcal{S}^{a_i}, s' \in \mathcal{S}^{a_j} \setminus s_j\}, 0), \tag{3.11}$$

$$r^3_{s_i,s_j} \triangleq \text{percentile}(\{|\rho_{s,s'}| : s \in \mathcal{S}^{a_i}, s' \in \mathcal{S}^{a_j} \setminus s_j\}, 25), \tag{3.12}$$

$$r^4_{s_i,s_j} \triangleq \text{percentile}(\{|\rho_{s,s'}| : s \in \mathcal{S}^{a_i}, s' \in \mathcal{S}^{a_j} \setminus s_j\}, 50), \tag{3.13}$$

$$r^5_{s_i,s_j} \triangleq \text{percentile}(\{|\rho_{s,s'}| : s \in \mathcal{S}^{a_i}, s' \in \mathcal{S}^{a_j} \setminus s_j\}, 75), \tag{3.14}$$

$$r^6_{s_i,s_j} \triangleq \text{percentile}(\{|\rho_{s,s'}| : s \in \mathcal{S}^{a_i}, s' \in \mathcal{S}^{a_j} \setminus s_j\}, 100), \tag{3.15}$$

where the function percentile(C, p) returns the p^{th} percentile of the items in the list C sorted in non-decreasing order.

We evaluate the performance of the aggregation strategies using the facility location function. In particular we utilize the six different aggregation strategies to compute the pairwise similarity measure for defining the facility location function. We test separately on three cell types: H1-hESC, K562, and GM12878. Similar to the previous experiment for comparing among different submodular objectives, we measure the performance of an aggregation strategy as how the valuation given by facility location function defined via the similarity matrix computed using this strategy correlates with the three proposed evaluation metrics. Following the same evaluation procedure we show the Spearman correlation measure for each aggregation strategy on each cell type and evaluation metric in Figure 3.8. We

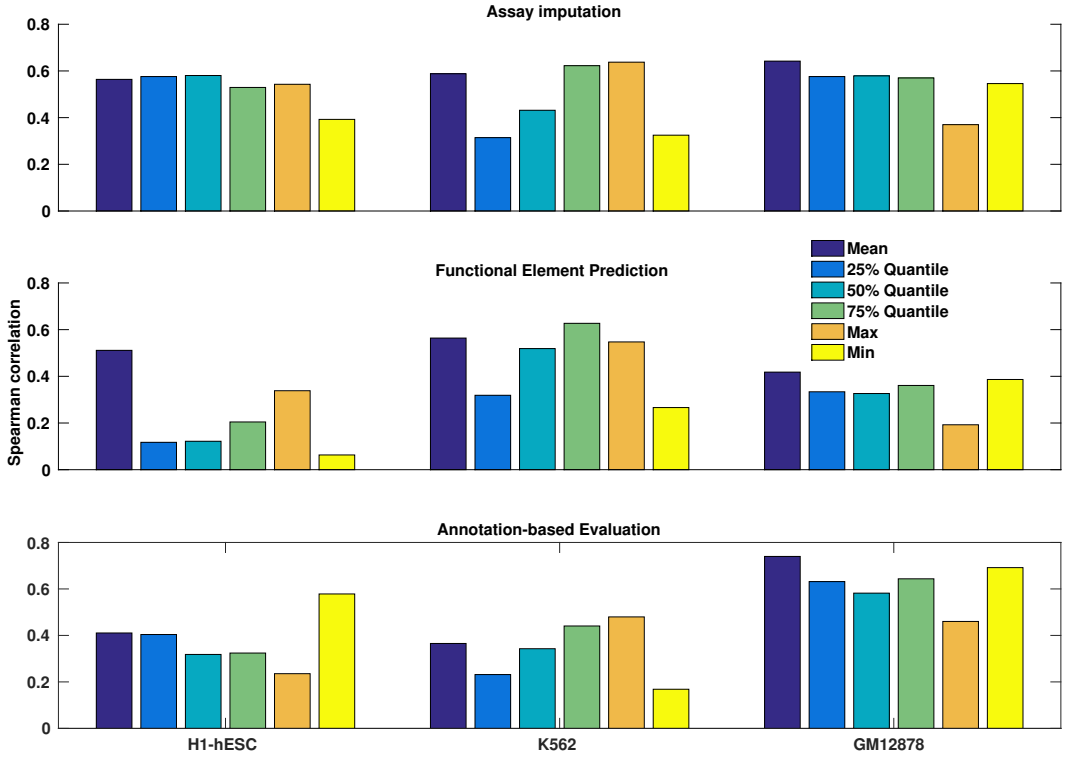


Figure 3.8: Comparison among various similarity aggregation strategies in terms of the Spearman correlation between the facility location function valuation instantiated by the similarity measure and the performance metrics.

observe that the aggregation strategies of taking the mean or 75^{th} percentile yield the best performance for most cell types and most evaluation metrics. Between these two strategies we observe that the aggregation by mean performs marginally better. Therefore we choose it as the strategy for computing the similarity between assay types in the propose approach SSA under the selection of future assays setting.

We also compared between two strategies for deriving the Pearson correlation. The first computes the correlation based on random samples of genomic positions, and the second on

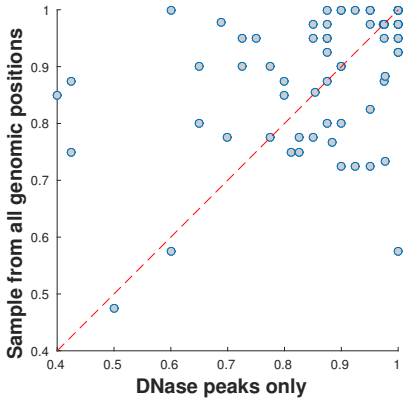


Figure 3.9: Scatter plot between the two variants of SSA with different genomic support for similarity computation. Each dot in the plot corresponds to the performance (measured as relative to an estimate of the performance on all possible panels) of the two variants of SSA for a selection budget ($k = 3, 4, 5, 6$) evaluated using a metric (assay imputation, function element prediction, and annotation-based evaluation) in a cell type (K562, GM12878, and H1-hESC). Its x- and y-values are the performance measure for the DNase peaks-based SSA and random genomic positions-based SSA, respectively.

the DNase peak positions only. We show the comparison between these two variants in the form of scatter plot in Figure 3.9. We observe that consistent and significant improvements over the random selection baseline are achieved by the variant of SSA using DNase peaks only. Between the two variants of SSA, it is hard to establish the superiority of one variant over the other according to Figure 3.9.

We lastly compared the effectiveness of using Pearson versus Spearman correlation for computing the similarity measure. We show the comparison of the performance between the Pearson correlation-based SSA and Spearman correlation-based SSA in Figure 3.10. We found that consistently better performance is achieved when the similarity is defined using the Pearson correlation.

These observations led us to choose our variant of facility location as the SSA's objective

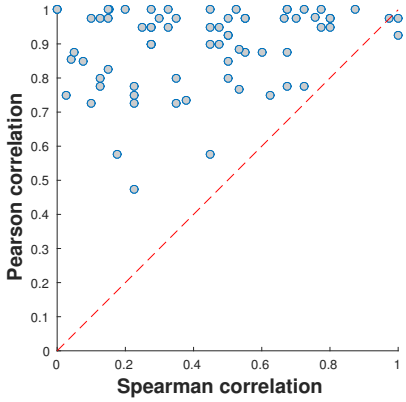


Figure 3.10: Scatter plot between the two variants of SSA with different correlation metrics as the similarity measure. Each dot in the plot corresponds to the performance (again measured as relative to an estimate of the performance on all possible panels) of the two variants of SSA for a selection budget evaluated using a metric in a cell type. Its x- and y-values are the performance measure for the Spearman correlation-based SSA and the Pearson correlation-based SSA, respectively.

function. As illustrated in Figure 3.11, such variant of the objective shows great correlation with all three evaluation metrics.

3.3.4 Panels chosen by SSA perform well on three evaluation metrics

In this section, we utilize the best SSA variant as discussed in the previous section and compare it to alternative panel selection approaches over the three evaluation metrics. As a baseline, we considered randomly selected panels of a given size. We also considered the panel of most frequently performed assays (Table A.1) as a good proxy for a likely data-driven choice. We found that the panels reported by SSA perform among the top few percent out of the space of all possible panels, and this high performance is consistent across panel sizes, evaluation cell types and performance metrics (Figure 3.12). We found that SSA also greatly outperforms the panel of most-frequently performed assay types. Indeed, this

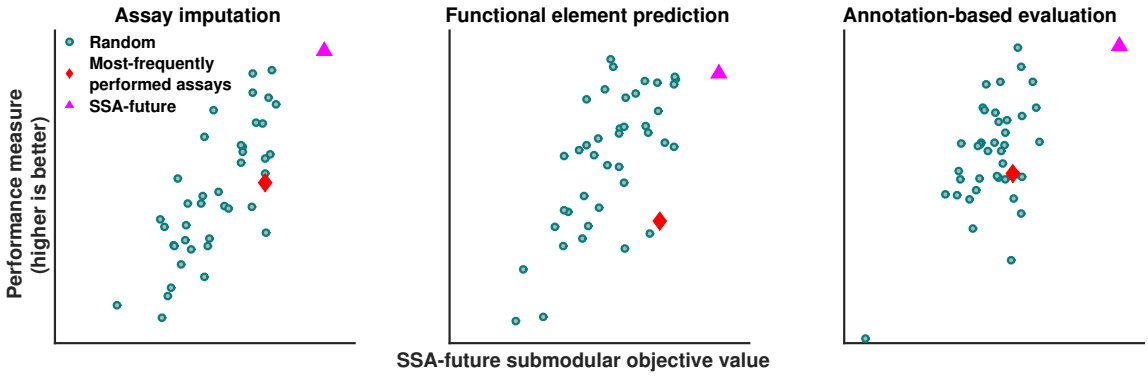
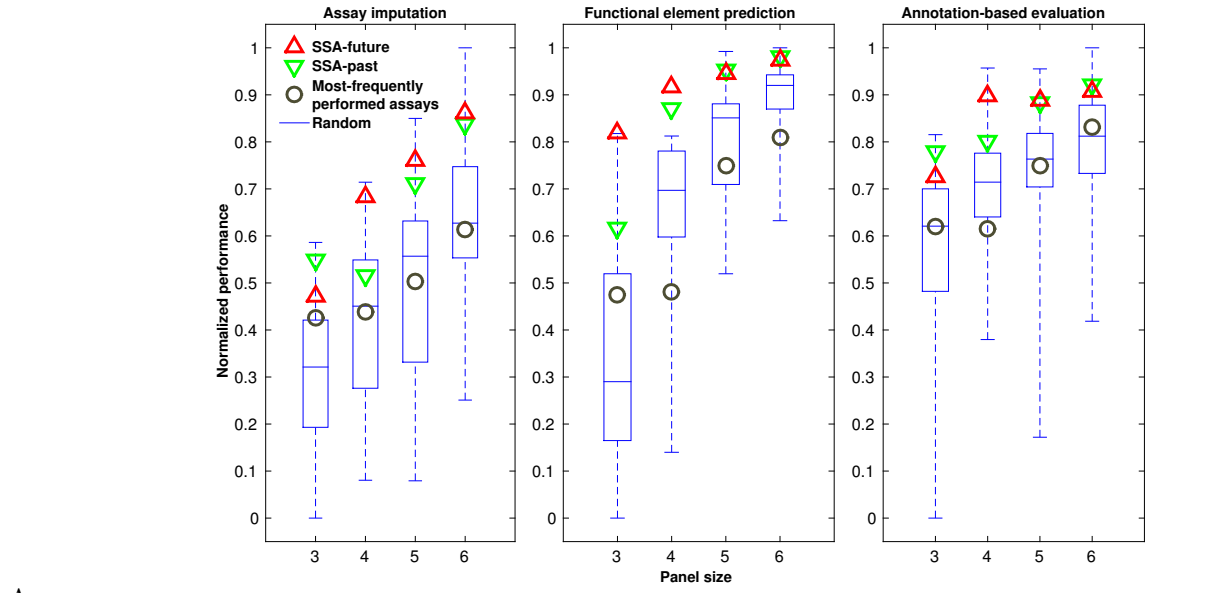


Figure 3.11: Relationship between the facility location objective function and evaluation metrics. Each dot corresponds to one of 40 randomly-chosen panels. Pink triangles indicate results from maximizing the SSA-future facility location function; red diamonds indicate the panel of most-frequently performed assay types (Table A.1). These results were computed in GM12878, using panels of four assay types.

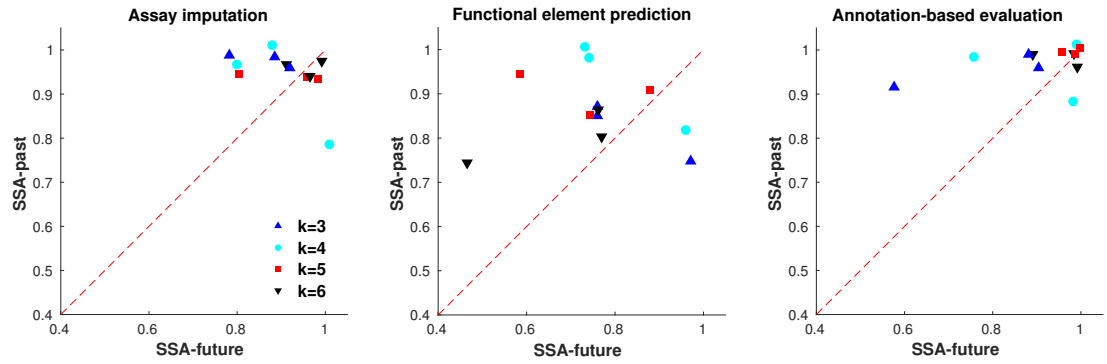
commonly-performed panel actually performs worse than the average panel in many cases, which may be a consequence of the fact that the most commonly-performed assay types measure broad marks of regulation, such as histone modifications and DNA accessibility, which do not have the specificity to identify pathway-specific elements. These results demonstrate quantitatively that panels chosen by SSA are effective when applied to their most common downstream tasks.

3.3.5 SSA can select a subset of performed assays as input to an expensive analysis

So far we have considered panel selection in the "future" setting, where a researcher is planning to experimentally perform a panel of assay types. Panel selection is sometimes also important in the "past" setting, where a researcher wishes to apply a computationally expensive analysis that cannot be efficiently applied to all assays together and therefore must be applied to a smaller panel. Importantly, in the selection of past assays case, a different panel can be selected for each cell type, based on the available data. To test SSA in the



A



B

Figure 3.12: Performance of panel selection strategies. (a) Boxplots show the distribution of evaluation metrics over 40 random panels on data from cell type GM12878. The panels of most-frequently performed assays are composed of the top k most frequent assay types available in our data set, where k is the size of the panel. Each evaluation metric is normalized to lie within $[0, 1]$ by subtracting the lowest value and dividing by the highest. (b) Scatter plot between the performance of SSA-past and SSA-future across cell types K562, GM12878, and H1-hESC. Each dot in the plot corresponds to the two variants of SSA for a panel size evaluated using a metric in a cell type. The performance is measured as the fraction of panels that perform worse than the SSA-chosen panel, estimated by comparing to 40 randomly-selected panels.

past setting, we used the same evaluation strategy as in the future setting, but using the source assays themselves to compute the similarity matrix. SSA performs consistently well according to these metrics, and it performs slightly better on some cell types in the past than the future setting due to the availability of this additional information (Figure 3.12B).

3.4 Discussion

In this chapter, we provide a submodular data summarization paradigm– SSA– for the assay panel selection problem. Moreover, we provide three downstream analysis techniques for automatically evaluating the quality of any panel of assay types. SSA is shown effective on selecting assay types from the ENCODE and Roadmap data sets. In particular, our method almost recapitulates the choice of histone modification assay types made by researchers. Moreover, our method proves to be empirically effective on all three evaluation metrics proposed here for both the future and the past setting.

It would be interesting in the future to apply the same submodular data summarization paradigm to select a set of informative and representative cell types to study. This cell type panel selection setting can be viewed as a dual variant of the assay panel selection problem. The methodologies proposed here for evaluating the quality of a panel of assay types (e.g., assay imputation, and annotation-based evaluation) may also be extended to assess the goodness a cell type panel selection approach.

Another direction of this work is to study assay selection in a more general form. Instead of forcing to choose some assay type from one cell type, an interesting scenario is to analyze how to arrange the order of assays to perform across assay types and cell types so that the most information about all cell types can be learned.

Chapter 4

CASE STUDY III: BATCH ACTIVE LEARNING

In our third case study, we aim to understand a bit better about the submodular functions for modeling the utility of data sets. Given the great success of the submodular data summarization paradigm on the speech data subset selection and the genomics assay panel selection problems, we study how to design the submodular utility function in a more principled way. We focus on the problem of data summarization for training machine learning (ML) classifiers. Here the goal is to identify a subset of the data so as to minimize any significant loss in training a machine learning classifier. The problem can be categorized into three scenarios depending whether the labels of the data are available to the data summarization algorithm:

1. Supervised setting: the selection algorithm has access to the labels of the training data.

2. Unsupervised setting: the selection algorithm does not use the labels for selecting data.

3. Active learning: Label queries are made on subsets of data in an iteratively manner.

Note that the categorization here is very similar to the speech data subset selection problem. In this chapter, we concentrate on both the supervised setting and the active learning setting. We first propose an approach to the supervised setting by connecting submodularity to likelihood functions of classifiers. Specifically, we express the utility set function for two simple classes of classifiers, the Naïve Bayes (NB) classifier and the Nearest Neighbor (NN) classifier utilizing submodularity. We identify two classes of submodular functions, *Naïve Bayes submodular* and *Nearest Neighbor submodular*, that naturally model maximum likelihood estimates over data subsets for both NB and NN classifiers. Data summarization for training either classifier is then performed as constrained submodular

maximization. The Naïve Bayes submodular function is a special case of the feature based submodular functions that have been successfully used for the speech data subset selection (see Chapter 2), while the Nearest Neighbor submodular function generalizes the class of facility location function [Mirchandani and Francis, 1990] that have also been previously successfully used for the speech data subset selection and the genomics assay panel selection problems (see Chapter 2 and 3).

Supervised data subset selection for the NN classifier, in particular, has great practical importance, since the NN classifier is non-parametric — i.e., the classifier must essentially memorize, and allocate storage for, the entire training set. The complexity of classifying one sample is dependent on the training set size, which can be expensive for large-scale applications. Our supervised data subset selection strategies reduce the training set size, and when the submodular function used to perform the subset selection is matched with the NN classifier, there is little performance loss even though the NN classifier has significantly less data to memorize.

Armed with the theoretical analysis of the utility functions, we next study the active learning setting by extending the proposed supervised submodular data summarization methods. We focus on the batch-mode active learning scenario, where there are multiple rounds of data selection, each of which selects a batch of data points whose label may be used to select the data points at future rounds. In fact, the problem of active learning has been studied by a number of authors. Among them, some have connected submodularity to active learning in various ways. Just to name a few, [Hoi et al., 2006] investigate the batch active learning problem for logistic regression, and connect this to submodular optimization. They model the utility of labeling a set of instances as the reduction on the Fisher information and show this utility function is monotone submodular. [Guillory and Bilmes, 2011] study the role of submodular functions in subset selection for very general simultaneously active and semi-supervised learning algorithms. Another thread of work [Guillory and Bilmes, 2010, Golovin and Krause, 2010, Cuong et al., 2010] provides a link between submodularity and active learning through notions of interactive and adaptive submodularity. In much of this work, they model version space reduction via adaptive submodular functions. While this focuses

on the fully adaptive setting, [Chen and Krause, 2013] extend this to the batch setting. Other algorithms having an active learning flavor include selecting the most informative set of items at every round [Settles, 2010], where the *informativeness* refers to utility of the items from the classifiers' point of view. The informativeness is often measured by the classifier's uncertainty (in the case of *uncertainty sampling*) [Lewis and Gale, 1994] or the variance (in the case of *Query by Committee*) [Seung et al., 1992]. It is interesting to note that some of these methods, e.g., uncertainty sampling, are special cases of adaptive submodular maximization [Cuong et al., 2010]. These techniques do not ideally capture the *representativeness* of the samples, a problem further aggravated in the batch setting. By *representativeness*, we mean how well a set of items covers the entire training set. This aspect is naturally modeled by density based methods [Nguyen and Smeulders, 2004]. In order to obtain the benefits of both classes of algorithms, several papers have combined both notions (informativeness and representativeness) in a single objective [Xu et al., 2003, Huang et al., 2010].

To address the batch active learning problem, we propose a novel multi-stage scheme we call *filtered active submodular selection* (FASS). At every round, as more labeled data becomes available, FASS uses an improved approximation to supervised data subset selection. We show how our framework naturally combines the notions of sample *informativeness* and, via submodularity, *representativeness*. We also show how our method is scalable relative to existing active learning techniques that attempt to combine these two notions. As we will see in Section 4.3, FASS significantly outperforms existing active learning baselines over a number of classifiers, including Naïve Bayes, Nearest Neighbor, Logistic Regression, and Deep Neural Networks.

4.1 Supervised Data Subset Selection

4.1.1 Naïve Bayes Classifier

We first consider the class of Naïve Bayes classification problems. Let $V = \{(x^i, y^i)\}_{i=1}^m$ be a set of training samples, where $x^i \in \mathcal{X}^d$ is a d-dimensional feature vector, and each feature

takes a value from the finite set $\mathcal{X}$; each sample's label $y^i \in \mathcal{Y}$ takes a value from the finite set $\mathcal{Y}$ of classes. The ground set V may be partitioned as $V = V^1 \cup V^2 \cup \cdots \cup V^{|\mathcal{Y}|}$, where V^y is the set of all samples in V with class label y. We write the jth dimension of a feature vector x as $x_j \in \mathcal{X}$. We denote the maximum likelihood (ML) estimate of the parameters θ of a Naïve Bayes model, given a set of training samples $S \subseteq V$, as $\theta(S)$, and also use $\theta_{x_j|y} = p(x_j|y)$ and $\theta_y = p(y)$. For simplicity, we first assume no smoothing occurs during estimation but this will be considered later. The ML parameter function $\theta(S)$ can be given as follows:

$$\theta_{x_j|y}(S) = \frac{m_{x_j,y}(S)}{m_y(S)}; \tag{4.1}$$

and

$$\theta_y(S) = \frac{m_y(S)}{|S|}, \tag{4.2}$$

where $m_{x_j,y}(S) = \sum_{i \in S} 1\{x_j^i = x_j \wedge y^i = y\}$ and $m_y(S) = \sum_{i \in S} 1\{y^i = y\}$. By definition, $m_{x_j,y}(S)$ counts the number of samples in S whose class label is $y \in \mathcal{Y}$ and whose jth dimension feature takes value $x_j \in \mathcal{X}$. Similarly, $m_y(S)$ counts the number of samples in S whose class label is $y \in \mathcal{Y}$. Both $m_{x_j,y}(S)$ and $m_y(S)$ are modular set functions, i.e., for any $S \subseteq V$, $m_{x_j,y}(S) = \sum_{s \in S} m_{x_j,y}(s)$ and $m_y(S) = \sum_{s \in S} m_y(s)$. Given the parameter function $\theta(S)$, we introduce the notion of *data log-likelihood set function* $\ell^{\text{NB}} : 2^V \to \mathbb{R}$ that maps from each set $S \subseteq V$ of training samples to a log likelihood evaluated on the whole data set V:

$$\ell^{\text{NB}}(S) = \sum_{i \in V} \log p(x^i, y^i; \theta(S)), \tag{4.3}$$

where $p(x^i, y^i; \theta(S))$ is the likelihood of the sample i parameterized by $\theta(S)$. $\ell^{\text{NB}}(S)$ acts as a utility set function for training a NB classifier. Under the Naïve Bayes assumption, we can express $\ell^{\text{NB}}(S)$ as follows:

$$\ell^{\text{NB}}(S) = \sum_{i \in V} \log p(x^i|y^i; \theta(S)) + \log p(y^i; \theta(S))$$

$$
\begin{aligned}
&= \sum_{i \in V} \sum_{j=1}^{d} \log p(x_j^i | y^i; \theta(S)) + \sum_{i \in V} \log p(y^i; \theta(S)) \\
&= \sum_{i \in V} \sum_{j=1}^{d} \log \theta_{x_j^i | y^i}(S) + \sum_{i \in V} \log \theta_{y^i}(S) \\
&= \sum_{i \in V} \sum_{j=1}^{d} \log \frac{m_{x_j^i, y^i}(S)}{m_{y^i}(S)} + \sum_{i \in V} \log \frac{m_{y^i}(S)}{|S|} \\
&= \underbrace{\sum_{j=1}^{d} \sum_{x_j \in \mathcal{X}} \sum_{y \in \mathcal{Y}} m_{x_j, y}(V) \log(m_{x_j, y}(S))}_{\text{term 1: } f_{\text{NB}}(S)} \\
&\quad \underbrace{- (d-1) \sum_{y \in \mathcal{Y}} m_y(V) \log(m_y(S))}_{\text{term 2}} \underbrace{- |V| \log |S|}_{\text{term 3}}.
\end{aligned}
$$

Since $m_{x_j,y}(V)$, $m_y(V)$, and $|V|$ are all independent of S, they can be treated as constants in $\ell^{\text{NB}}(S)$. Then the first, second, and third terms of $\ell^{\text{NB}}(S)$ are in the form of a sum of concave over modular functions, hence they are all monotone submodular [Lin and Bilmes, 2011]. As a result, $\ell^{\text{NB}}(S)$ is in the form of difference of submodular functions, and the underlying optimization problem becomes a difference of submodular (DS) optimization (see Problem 5):

$$\max_{|S|=k} \ell^{\text{NB}}(S). \tag{4.4}$$

While there are scalable heuristics that work well in practice to minimize a difference of submodular functions [Narasimhan and Bilmes, 2005, Iyer et al., 2014], these techniques lack worst-case guarantees since the underlying problem is hard to optimize. Fortunately, the second and the third term of $\ell^{\text{NB}}(S)$ become constants when we enforce a set of inconsequential constraints. We call the first term (the only active term in a transformed optimization problem) the *Naïve Bayes submodular function*:

$$f_{\text{NB}}(S) = \sum_{j=1}^{d} \sum_{x_j \in \mathcal{X}} \sum_{y \in \mathcal{Y}} m_{x_j, y}(V) \log m_{x_j, y}(S). \tag{4.5}$$

Given the equality constraint $|S| = k$, the third term in $\ell^{\text{NB}}(S)$ becomes a constant in Problem 4.4. Furthermore, we make an assumption that the selected set should be balanced, which makes the second term also a constant. In particular, we say that a set S is *balanced* if S maintains the same distribution over the class labels as the whole data set. A set S of size k is balanced if the proportion of each class label in the set S is the same as the whole data set V, i.e., $|S \cap V^y| = k\frac{|V^y|}{|V|}$ for any $y \in \mathcal{Y}$ for some k. We assume that k is chosen such that $k\frac{|V^y|}{|V|}$ is an integer for all $y \in \mathcal{Y}$, and if not then we round $k\frac{|V^y|}{|V|}$ to the closest integer. With balance enforced and $|S| = k$, we have that $m_y(S) = |S \cap V^y| = k\frac{|V^y|}{|V|}$ for all $y \in \mathcal{Y}$. Therefore, term 2 of $\ell^{\text{NB}}(S)$ also becomes a constant. Let $\mathcal{M}(V, \mathcal{I})$ be a partition matroid using the partition $\{V^y\}_{y\in\mathcal{Y}}$, where the set of bases $\mathcal{B}(\mathcal{M})$ is defined as $\mathcal{B}(\mathcal{M}) = \{S \subseteq V : |S \cap V^y| = k\frac{|V^y|}{|V|}, \forall y \in \mathcal{Y}\}$. Thus, a set S of size k being balanced is equivalent to S being a base of the matroid $\mathcal{M}$, i.e., $S \in \mathcal{B}(\mathcal{M})$. Since the second and the third term of $\ell^{\text{NB}}(S)$ are constants given the constraint $S \in \mathcal{B}(\mathcal{M})$, the above optimization problem is equivalent to:

$$\max_{S\in\mathcal{B}(\mathcal{M})} f_{\text{NB}}(S). \tag{4.6}$$

It becomes maximizing a submodular function under a matroid constraint (see Problem 3), for which one may efficiently, scalably, and approximately solve using the lazy greedy algorithm [Fisher et al., 1978]. This formulation asks for a small training data subset S such that the likelihood of the ML parameters $\theta(S)$ is large on the *entire* data set V, and the following Theorem offers perspective in terms of the KL-divergence.

Theorem 10. *Let* $D_{KL}(p(x, y; \theta(V))||p(x, y; \theta(S))) \triangleq \sum_{x\in\mathcal{X}^d}\sum_{y\in\mathcal{Y}} p(x, y; \theta(V)) \log \frac{p(x,y;\theta(V))}{p(x,y;\theta(S))}$ *be the KL-divergence between* $p(x, y; \theta(V))$ *and* $p(x, y; \theta(S))$*, where* $p(x, y; \theta(S))$ *is the maximum likelihood estimate of the joint distribution given a data set* S*. Under the Naïve Bayes assumption, Problem 4.4 is equivalent to*

$$\min_{|S|=k} D_{KL}(p(x, y; \theta(V))||p(x, y; \theta(S))). \tag{4.7}$$

Proof. We derive D_{KL} as follows:

$$
\begin{aligned}
&D_{KL}(p(x,y;\theta(V))||p(x,y;\theta(S))) \\
&= -\sum_{x_1,\ldots,x_d\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x_1,\ldots,x_d,y;\theta(V))\log p(x_1,\ldots,x_d,y;\theta(S)) \\
&+ \underbrace{\sum_{x_1,\ldots,x_d\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x_1,\ldots,x_d,y;\theta(V))\log p(x_1,\ldots,x_d,y;\theta(V))}_{\text{constant: } C} \\
&= -\sum_{x_1,\ldots,x_d\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x_1,\ldots,x_d,y;\theta(V))\sum_{j=1}^{d}\log p(x_j|y;\theta(S)) \\
&- \sum_{x_1,\ldots,x_d\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x_1,\ldots,x_d,y;\theta(V))\log p(y;\theta(S)) + C \\
&= -\sum_{j=1}^{d}\sum_{x_j\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x_j,y;\theta(V))\log p(x_j|y;\theta(S)) \\
&- \sum_{y\in\mathcal{Y}} p(y;\theta(V))\log p(y;\theta(S)) + C \\
&= -\sum_{j=1}^{d}\sum_{x_j\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \frac{m_{x,y}(V)}{|V|}\log\frac{m_{x,y}(S)}{m_y(S)} - \sum_{y\in\mathcal{Y}}\frac{m_y(V)}{|V|}\log\frac{m_y(S)}{|S|} + C \\
&= \frac{1}{|V|}\{-\sum_{j=1}^{d}\sum_{x_j\in\mathcal{X}}\sum_{y\in\mathcal{Y}} m_{x_j,y}(V)\log m_{x_j,y}(S) + |V|\log|S| \\
&+ (d-1)\sum_{y\in\mathcal{Y}} m_y(V)\log m_y(S)\} + C \\
&= -\frac{1}{|V|}\ell^{\mathrm{NB}}(S) + C
\end{aligned}
$$

Therefore, Problem 4.7 is equivalent Problem 4.4 as shown below:

$$D_{KL}(p(x,y;\theta(V))||p(x,y;\theta(S))) = -\frac{1}{|V|}\ell^{\mathrm{NB}}(S) + C, \tag{4.8}$$

□

We next point out connections between the Naïve Bayes submodular function f_{NB} and the class of feature-based submodular functions f_{fea} that have effectively been applied in the

speech data subset selection task in Chapter 2. Recall that the feature-based function is defined in the following form:

$$f_{\text{fea}}(S) = \sum_{u \in U} w_u g(c_u(S)),$$

where g is a concave function, U is a set of "features", $\{w_u\}_{u \in U}$ is a set of non-negative weights for each feature $u \in U$, and $c_u(S) = \sum_{s \in S} c_u(s)$ is a non-negative modular score for feature $u \in U$ in S, with $c_u(s)$ measuring the degree to which item s "possesses" feature u. Defining U as the set of all possible input-label pairs, i.e., $U = \{(x_j, y) | x_j \in \mathcal{X}, y \in \mathcal{Y}, j = 1, \ldots, d\}$, the weight for each feature as $w_u = m_u(V)$, the concave function as $g(x) = \log x$, and the modular score as $c_u(S) = m_u(S)$, f_{NB} is then an instance of a feature-based function. Maximizing f_{NB} chooses data that has diverse coverage in the set of features $\mathcal{U}$ where the desired coverage for each feature $u \in \mathcal{U}$ is controlled by its weight $w_u = m_u(V)$.

Laplace smoothing: Without any smoothing, it may be that the Naïve Bayes submodular function f_{NB} is undefined at $S = \emptyset$. Smoothing not only fixes this issue, it is naturally incorporated into the submodular framework. Given a Laplace smoothing parameter $\alpha > 0$, for a set $S \subseteq V$, the Laplace smoothed ML estimated parameters become

$$\theta^{\alpha}_{x_j|y}(S) = \frac{m_{x_j,y}(S) + \alpha}{m_y(S) + \alpha|\mathcal{X}|}; \tag{4.9}$$

$$\theta^{\alpha}_{y}(S) = \frac{m_y(S) + \alpha}{|S| + \alpha|\mathcal{Y}|}. \tag{4.10}$$

We may formulate this similar to Problem 4.6, where the objective is slightly modified from before. First, define a slightly expanded ground set $V' = V \cup \{v'\}$ where v' is a pseudo-sample that has the property $m_{x_j,y}(v') = \alpha, \forall x_j \in \mathcal{X}, y \in \mathcal{Y}, j = 1, \ldots, d$. We next define a function $f'_{\text{NB}(\alpha)} : 2^{V'} \to \mathbb{R}_+$ as:

$$f'_{\text{NB}(\alpha)}(S) = \sum_{j=1}^{d} \sum_{x_j \in \mathcal{X}} \sum_{y \in \mathcal{Y}} m_{x_j,y}(V) \log(m_{x_j,y}(S)) \tag{4.11}$$

For any $S \subseteq V$, $f'_{\mathrm{NB}(\alpha)}(S \cup \{v'\})$ represents the Laplace-smoothed ML objective. From this, we obtain a normalized monotone submodular function $f_{\mathrm{NB}(\alpha)} : 2^V \to \mathbb{R}_+$ where $f_{\mathrm{NB}(\alpha)}(S) = f'_{\mathrm{NB}(\alpha)}(S \cup \{v'\}) - f'_{\mathrm{NB}(\alpha)}(\{v'\})$ whose score is the Laplace-smoothed ML estimate minus a constant.

Naïve Bayes in Text Classification: Next, we consider the problem of text classification, where Naïve Bayes classifier under the bag-of-words model often performs very well. In this context, $V = \{(D^i, y^i)\}_{i \in V}$ is a set of document-label pairs, $\mathcal{Y}$ is a set of document labels (e.g., topics), each document D^i is assigned to only one label $y^i \in \mathcal{Y}$. We represent each document D^i as a bag of words $D^i = \{w_j\}_{j=1}^{n_i}$, where n_i is the number of words in the document, and each word w_j is taken from a vocabulary $\mathcal{W}$. Let $c_y(S) = \sum_{i \in S} 1\{y^i = y\} n_i$ counts the number of words in the set S of documents labeled as y. $c_y(S)$ is a modular function for each label $y \in \mathcal{Y}$. We also define $m_{w,y}(S) = \sum_{i \in S} m_{w,y}(i)$ where $m_{w,y}(i) = \sum_{w' \in D_i} 1\{w' = w \wedge y^i = y\}$. By definition, $m_{w,y}(S)$ counts the number of occurrences of $w \in \mathcal{W}$ in the subset S of documents that are labeled as y. Following the same spirit, we define the data log likelihood function $\ell^{\mathrm{NB}}_{\mathrm{text}}(S)$ with the following form:

$$\ell^{\mathrm{NB}}_{\mathrm{text}}(S) = \sum_{i \in V} \log p(D^i, y^i; \theta(S)). \tag{4.12}$$

Under the Naïve Bayes assumption, we can simplify it as follows:

$$\begin{aligned}
\ell^{\mathrm{NB}}_{\mathrm{text}}(S) &= \sum_{i \in V} \log p(D^i | y^i; \theta(S)) + \sum_{i \in V} \log p(y^i; \theta(S)) \\
&= \sum_{i \in V} \sum_{w \in D^i} \log p(w | y^i; \theta(S)) + \sum_{i \in V} \log p(y^i; \theta(S)) \\
&= \sum_{i \in V} \sum_{w \in D^i} \log \frac{m_{w,y^i}(S)}{c_{y^i}(S)} + \sum_{i \in V} \log \frac{m_{y^i}(S)}{|S|} \\
&= \underbrace{\sum_{w \in \mathcal{W}} \sum_{y \in \mathcal{Y}} m_{w,y}(V) \log m_{w,y}(S)}_{\text{term 1:} f_{\text{NB-text}}} - \underbrace{|V| \log |S|}_{\text{term 2}} \\
&\quad - \underbrace{\sum_{y \in \mathcal{Y}} c_y(V) \log c_y(S)}_{\text{term 3}} + \underbrace{\sum_{y \in \mathcal{Y}} m_y(V) \log m_y(S)}_{\text{term 4}}
\end{aligned}$$

The data log likelihood function $\ell^{\text{NB}}_{\text{text}}(S)$ is again in the form of difference of submodular functions. By enforcing the chosen set S to be of fixed size and balanced, term 2 and 4 can be handled as constants. Furthermore, term 3 can also be a constant if each document has the same length d, i.e., $n_i = d$ for all $i \in V$. This can be a reasonable assumption to make, since one can always normalize the word counts for each document such that each document has constant length with potentially fractional word counts, and more importantly, it has been reported in [Nigam et al., 2000] that better bag-of-words NB model may be trained if the training documents are word-count normalized. We formulate the supervised data subset selection for text classification problem under the bag-of-words Naïve Bayes model as follows:

$$\max_{S \in \mathcal{B}(\mathcal{M})} \ell^{\text{NB}}_{\text{text}}(S), \tag{4.13}$$

which can be equivalently transformed to the following:

$$\max_{S \in \mathcal{B}(\mathcal{M})} f_{\text{NB-text}}(S), \tag{4.14}$$

where

$$f_{\text{NB-text}}(S) = \sum_{w \in \mathcal{W}} \sum_{y \in \mathcal{Y}} m_{w,y}(V) \log m_{w,y}(S). \tag{4.15}$$

The same Laplace smoothing technique can also be naturally incorporated into this framework and make $f_{\text{NB-text}}$ well-defined at $S = \emptyset$.

4.1.2 Nearest Neighbor Classifier

In this section, we consider (non-parametric) Nearest Neighbor (NN) classifiers and formulate supervised data subset selection problem as constrained submodular maximization. Given a set of training samples $V = \{(x^i, y^i)\}_{i=1}^m \subseteq \mathcal{X} \times \mathcal{Y}$ and a similarity function $w : V \times V \to \mathbb{R}^+$ which measures the similarity between any pair of data instances in feature space, the NN classifier simply classifies $x \in \mathcal{X}$ based on its closest training sample. The similarity between training sample pairs i and j is given as $w(i, j) = d - \|x^i - x^j\|_2^2 \geq 0$, where

$d = \max_{v \in V, v' \in V} \|x^v - x^{v'}\|_2^2$ is the maximum pairwise distance. Though extremely simple, the NN classifier has been applied on a number of machine learning tasks, including hand-written digit recognition, text categorization, object recognition, etc [Bhatia et al., 2010, Boiman et al., 2008]. For a NN classifier, no model needs to be "learnt" as nearly all the computation takes place at the classification stage. The complexity of classifying a sample can be expensive and is dependent on the number of training samples. A way to alleviate this problem is to reduce the training set size ideally without losing performance, a problem well suited to supervised data subset selection.

Similar to the NB classifier analysis, we consider a data log-likelihood set function $\ell^{\text{NN}} : 2^V \to \mathbb{R}$ that maps each subset $S \subseteq V$ to a log likelihood score on the whole set V:

$$\ell^{\text{NN}}(S) = \sum_{i \in V} \log p(x^i | y^i; \theta(S)) + \sum_{i \in V} \log p(y^i; \theta(S)),$$

where $p(x^i|y^i;\theta(S))$ and $p(y^i;\theta(S))$ are the generative likelihood and the prior likelihood of the sample $i \in V$ given by $\theta(S)$. The idea of the data subset selection is to select a small sized set S so that $\ell^{\text{NN}}(S)$ is maximized. As in the NB scenario, we express the prior as $p(y^i;\theta(S)) = \frac{m_{y^i}(S)}{|S|}$. The key question regarding the function ℓ^{NN} is how to appropriately characterize the generative likelihood function $p(x^i|y^i;\theta(S))$ so that it is of a simple form leading ℓ^{NN} to be submodular and also maps well to the NN classifier. To this end, we assume the following:

1. $p(x^i|y^i;\theta(S))$ is determined only by the sample j in S that is with label y^i and is closest to i, i.e., $j \in \text{argmax}_{s \in S \cap V^{y^i}} w(i,s)$;

2. The generative likelihood is expressed as $p(x^i|y^i;\theta(S)) = ce^{-\|x^i - x^j\|_2^2} = ce^{w(i,j)-d} = c'e^{w(i,j)} = c' \exp\left(\max_{s \in S \cap V^{y^i}} w(i,s)\right)$, where c and c' are constants.

We express the generative log-likelihood as $\log p(x^i|y^i;\theta(S)) = \log c' + \max_{s \in S \cap V^{y^i}} w(i,s)$

yielding:

$$\ell^{\text{NN}}(S) = \underbrace{\sum_{y\in\mathcal{Y}}\sum_{i\in V^y}\max_{s\in S\cap V^y} w(i,s)}_{\text{term 1:}f_{\text{NN}}} + \underbrace{\sum_{y\in\mathcal{Y}} m_y(V)\log m_y(S)}_{\text{term 2}}$$
$$- \underbrace{|V|\log|S|}_{\text{term 3}} + \underbrace{C}_{\text{constant}}.$$

The first term is the *Nearest Neighbor submodular function*:

$$f_{\text{NN}}(S) = \sum_{y\in\mathcal{Y}}\sum_{i\in V^y}\max_{s\in S\cap V^y} w(i,s). \tag{4.16}$$

Similar to the NB case, $\ell^{\text{NN}}(S)$ is a difference of submodular functions. In a manner similar to the NB classifier, we assume that the selected set is balanced and of fixed size k. The second and the third term of ℓ^{NN} are treated as constants. Hence, the problem

$$\max_{S:|S|=k} \ell^{\text{NN}}(S) \tag{4.17}$$

is equivalently expressed as constrained submodular maximization:

$$\max_{S\in\mathcal{B}(\mathcal{M})} f_{\text{NN}}(S). \tag{4.18}$$

Connection to facility location function: Next we show f_{NN}'s connections to the *facility location* function. Recall that the facility location function has the following form:

$$f_{\text{fac}}(S) = \sum_{i\in V}\max_{j\in S} w(i,j). \tag{4.19}$$

Facility location function is often used to identify representative instances from a big collection of items. Furthermore, we have already shown that the facility location is very effective in both the speech data subset selection and the genomics assay panel selection problems. Moreover, the facility location function has been used as the data summarization objective for a number of tasks by a number of authors [Zheng et al., 2014, Lin and Bilmes, 2011, Gomes and Krause, 2010]. Sharing very similar definitions, the facility location function f_{fac} is in fact a special case of f_{NN}, when all items in V take the same class labels, or equivalently, $|\mathcal{Y}| = 1$. Given its resemblance to f_{NN}, the facility location function should

naturally model the utility of data sets for training NN classifiers, although it was originally designed to measure the representativeness of each set S about the whole set V. Also, f_{NN} can be written as a sum of facility location functions since $f_{\text{NN}}(S) = \sum_{y\in\mathcal{Y}} f^y_{\text{fac}}(S \cap V^y)$ with $f^y_{\text{fac}}(S) = \sum_{i\in V^y} \max_{j\in S} w(i,j)$ a facility location with ground set V^y. As far as we know, we are the first to 1) connect the facility location function to the utility of training NN classifiers, and 2) to show that the utility of a set for training NN classifiers is measured by its representativeness about the data partition for each class.

Extension to k-Nearest Neighbor classifiers: Next, we extend the analysis of the likelihood function to the k-Nearest Neighbor classifiers for $k > 1$. Consider a data log-likelihood set function $\ell^{\text{kNN}} : 2^V \to \mathbb{R}$ that maps from each subset $S \subseteq V$ to a log-likelihood score of the k-NN classifier on the whole data set V:

$$\ell^{\text{kNN}} = \sum_{i\in V} \log p(x^i|y^i;\theta(S)) + \sum_{i\in V} \log p(y^i;\theta(S)). \tag{4.20}$$

It is straightforward to write the prior likelihood as $p(y^i;\theta(S)) = \frac{m_{y^i}(S)}{|S|}$. The remaining question is how to characterize the generative likelihood function $p(x^i|y^i;\theta(S))$ given a set of items $S \subseteq V$. In the context of a k-NN classifier, we assume the followings:

1. The similarity $w(i,j)$ between any pair of items i and j is computed as $w(i,j) = d - \|x^i - x^j\|_2^2$, where $d = \max_{i\in V, j\in V} \|x^i - x^j\|_2^2$ is the maximum pairwise Euclidean distance.

2. $p(x^i|y^i;\theta(S))$ is determined by the set of k samples $T \subseteq S$ that are with label y^i and are closest to i, i.e., $T \in \arg\max_{|T|=k;T\subseteq S\cap V^{y^i}} \sum_{t\in T} w(t,i)$, (here we assume $|S \cap V^{y^i}| \geq k$)

3. The generative likelihood is parameterized as

$$p(x^i|y^i;\theta(S)) \propto \prod_{t\in T} e^{-\|x^i - x^t\|_2^2} \tag{4.21}$$

$$= c \prod_{t\in T} e^{w_{i,t}-d} \tag{4.22}$$

$$= c' \exp\{\max_{T \subseteq S \cap V^{y^i}; |T|=k} \sum_{t \in T} w(i,t)\} \tag{4.23}$$

Using this, we derive the log generative likelihood as

$$\log p(x^i | y^i; \theta(S)) = \log c' + \max_{T \subseteq S \cap V^{y^i}; |T|=k} \sum_{t \in T} w(i,t),$$

leading to the following:

$$\ell^{\text{kNN}}(S) = \underbrace{\sum_{y \in \mathcal{Y}} \sum_{i \in V^y} \max_{T \subseteq S \cap V^y; |T|=k} \sum_{t \in T} w(i,t)}_{\text{term 1: } f_{\text{k-NN}}} + \underbrace{\sum_{y \in \mathcal{Y}} m_y(V) \log m_y(S)}_{\text{term 2}} - \underbrace{|V| \log |S|}_{\text{term 3}} + \underbrace{C}_{\text{constant}} .$$

Given the same assumption about the chosen set to be balanced, the first term is the only effective term in the transformed optimization problem. We call the first term *k-Nearest Neighbor submodular function* with the form $f_{\text{kNN}}(S) = \sum_{y \in \mathcal{Y}} \sum_{i \in V^y} \max_{T \subseteq S \cap V^y; |T|=k} \sum_{t \in T} w(i,t)$, which interestingly turns out to be in form of the weighted matroid rank functions as defined in [Shioura, 2009].

4.2 Batch Active Learning

We next extend the results of supervised data subset selection to the batch active learning setting, where we incrementally obtain labels. We define a multistage *adaptive active learning* framework, where selection adapts to the labels previously obtained. Moreover, we show how we can naturally combine the notions of *representativeness* and *information* by filtering. Here, we focus on uncertainty sampling (defined later) to represent *information*, but our methods can extend to other strategies, such as Query by Committee, as well.

In the batch active learning setting, the algorithm iteratively selects a set of B unlabeled instances to label at every round, and this is done for T rounds. Later rounds get to use the labels previously selected, so this is an adaptive strategy, but within each batch all labels

Algorithm 6: Filtered Active Submodular Selection

Input: $\mathcal{U}$, T, B, $\{\beta_t\}_{t=1}^{T}$, Starting set of labels $\mathcal{L}$

for $t = 1, \cdots, T$ **do**

  Train the classifier using the labeled set $\mathcal{L}$, and derive the uncertainty scores δ^t;

  $\mathcal{U}^t \in \text{argmax}_{U \subseteq \mathcal{U} \setminus \mathcal{L}; |U| = \beta_t} \sum_{u \in U} \delta_u^t$;

  Obtain the most probable labels as the hypothesized labels $\{\hat{y}_u\}_{u \in \mathcal{U}^t}$.

  Instantiate $\hat{f}_t : 2^{\mathcal{U}^t} \rightarrow \mathbb{R}_+$ on $\{\hat{y}_u\}_{u \in \mathcal{U}^t}$ and $\mathcal{U}^t$;

  Find $L^t \in \text{argmax}_{|S|=B; S \subseteq \mathcal{U}^t} \hat{f}_t(S)$.

  $\mathcal{L} = \mathcal{L} \cup L^t$.

end for

are selected simultaneously without mutual knowledge of or interaction with each other. In the end, we obtain $k = BT$ labeled instances. We denote the set of unlabeled instances as $\mathcal{U}$, and the goal is to select a labeled set $\mathcal{L}$ such that $|\mathcal{L}| = k$. A common strategy is *uncertainty sampling*, where the B most uncertain examples (from the current classifiers perspective) are chosen for labeling [Lewis and Gale, 1994] at every round. Given a round t and a set of labeled items $\mathcal{L}$, let $\delta_u^t \geq 0$ be the uncertainty score for an example $u \in \mathcal{U} \setminus \mathcal{L}$. The uncertainty sampling approach, then, simply selects $S \in \max_{S' \subseteq \mathcal{U} \setminus \mathcal{L}; |S'| = B} \sum_{u \in S'} \delta_u^t$, and adds these to the labeled set $\mathcal{L}$. The drawback of this approach is that it fails to model the interactions between samples, i.e., labeling one sample could often affect the utility of labeling another. Simply choosing the most uncertain samples might lead to a selected set with high redundancy. A better strategy would choose a diverse set of samples from amongst those that the currently trained model is most uncertain about.

To this end, we propose a multi-stage batch active learning scheme called *filtered active submodular selection (*FASS*)*. This algorithm (see Alg. 9) attempts to solve the original data subset selection problem of maximizing a submodular function f (i.e., either the Naïve Bayes submodular function f_{NB}, or the NN submodular function f_{NN}) in an iterative manner.

At every round t, we first filter out data samples that the current model is certain about, and preserve a candidate set of β_t ($\beta_t \geq B$) most uncertain samples. Specifically, we find a solution to $\mathcal{U}^t \in \text{argmax}_{U' \subseteq \mathcal{U} \setminus \mathcal{L}; |U'| = \beta_t} \sum_{u \in U'} \delta_u^t$. Since we do not know the labels of the items in $\mathcal{U}^t$, we use the most probable prediction (based on the current classifier) $\hat{y}_u$ for each item $u \in \mathcal{U}^t$ as its hypothesized label. We then instantiate an appropriate submodular objective $\hat{f}_t : 2^{\mathcal{U}^t} \rightarrow \mathbb{R}_+$, which has essentially the same form as f, except that it is defined on the ground set $\mathcal{U}^t$, and uses the hypothesized labels $\hat{y}_u, \forall u \in \mathcal{U}^t$. We then solve the optimization problem

$$\max_{|S|=B; S \subseteq \mathcal{U}^t} \hat{f}_t(S). \tag{4.24}$$

The scheme of FASS is fully characterized by the choice of the monotone submodular objective f, the scaling parameters $\{\beta_t\}_{t=1}^T$ and the classifier. Given a classifier, better performance of FASS is expected when the submodular objective f matches the utility function for training the classifier. Hence in the case of the NB classifier, we use f_{NB} as the submodular function, while in the case of NN classifier, we can use f_{NN} or f_{fac}. In general however, we can use any submodular function f in our framework. In Section 4.3, we show that FASS with f_{NB} yields superior performance in the case of NB classifiers, while FASS with f_{fac} or f_{NN} performs better on NN classifiers.

Next, we discuss the scaling parameters $\{\beta_t\}_{t=1}^T$ for FASS. For any round t, β_t is the size of the candidate set $\mathcal{U}^t$ and controls the trade-off between the criteria of uncertainty and the submodular objective f. If $\beta_t = B, \forall t$, the selected set is chosen only accounting for the uncertainty scores, and FASS is reduced to the uncertainty sampling approach. When $\beta_t = |\mathcal{U} \setminus \mathcal{L}|$, the selected set does not account for the uncertainty scores, and is solely chosen by the submodular objective f. In our experiments, we set the scaling parameters at each round as constant β, i.e., $\beta_t = \beta, \forall t$. Choice of β affects the time and memory complexity of an instance of FASS scheme. It becomes significant when f is a graph-based submodular function, e.g., f_{NN} and f_{fac}. The time and memory complexity for constructing the similarity graph grows quadratically with β. Fortunately, we empirically observe that FASS often

performs rather well for small values of β ($\beta \ll |\mathcal{U}|$). As a result, FASS can easily scale to extremely large data sets. Thanks to our uncertainty sampling based filtering and data selection via submodular maximization, we naturally incorporate notions both of *information* and *representativeness*. Moreover, thanks to the greedy algorithm for maximization, as well as the prefiltering we perform, our approach can easily scale to large real-world machine learning problems.

We also point out that FASS subsumes the submodular active learning framework in [Hoi et al., 2006] as it is a special case of FASS with $\beta_t = |\mathcal{U} \setminus \mathcal{L}|$ and f being chosen as the *Fisher information submodular function* f_{fs} defined as

$$f_{\text{fs}}(S) = \frac{1}{c}\sum_{i \in \mathcal{U}} \pi_i(1-\pi_i) - \sum_{i \notin S} \frac{\pi_i(1-\pi_i)}{c + \sum_{j \in S} \pi_j(1-\pi_j)(x_i^T x_j)^2},$$

where $c > 0$, π_i is the posterior probability of a sample i, and x_i is the feature representation for i. It is shown in [Hoi et al., 2006] that f_{fs} is normalized monotone submodular. Their proof is done by using the diminishing return definition of submodularity, i.e., by proving $f(a|A) \geq f(a|B)$ for any $a \in V$ and $A \subset B \subseteq V$. Furthermore, they show that f_{fs} approximates the utility of a given set S by how much it reduces the Fisher information for logistic regression classifier. We compare our method with f_{fs} in the experiments.

4.3 Experimental Results

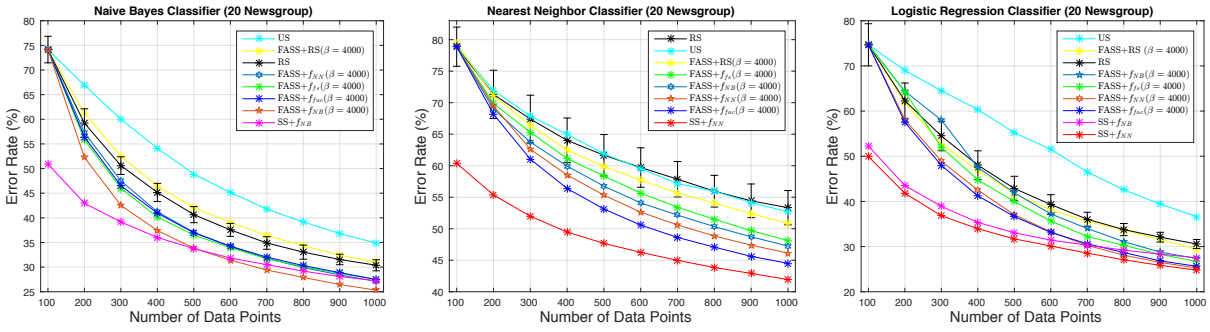
We empirically evaluate the proposed framework on the supervised data subset selection problem and the batch active learning problem. In the set of experiments, we wish to address the following: 1) How Eqn 4.6 and 4.18 perform on the supervised data subset selection for NB and NN classifier, respectively; 2) How FASS performs on active learning under various choices of f and β; and 3) How well the proposed framework extend to other classifiers, including Logistic Regression (LR) and Deep Neural Networks (DNN). In our experiments, we evaluate on two separate tasks: 1) text categorization, where we tested three classifiers: NB, NN, and LR; and 2) handwritten digit recognition, where we evaluated NN and DNN classifiers.

4.3.1 Text Categorization Experiments

Experimental setup: We evaluate text categorization on the 20 Newsgroups data set [1], which consists of 18774 articles divided almost evenly among 20 different UseNet discussion groups [Lang, 1995]. The goal is to classify an article into one newsgroup (of twenty) to which it was posted. For each instance of the experiment, we randomly split $\frac{2}{3}$ and $\frac{1}{3}$ of the whole data set as the training and test samples. Each subset selection strategy is applied to sub-select the training samples and then train a classifier. We report its classification error rate on the test set averaged over 20 instances of random data splits as the performance for each subset selection strategy. For batch active learning experiments, we first randomly label $B = 100$ samples, on which we train a classifier as the initial model. In each iteration, additional B unlabeled examples are selected for labeling to update the model. We evaluate for $T = 10$ iterations ending with a total of $k = 1000$ labeled examples. Under the least confident criterion, in each iteration t, we compute the uncertainty score δ_u^t of a sample u as $\delta_u^t = 1 - p(\hat{y}_u|x_u)$, where $\hat{y}_u$ is the most probable prediction of the sample u given by the currently trained model, i.e., $\hat{y}_u \in \text{argmax}_{y \in \mathcal{Y}} p(y|x_u)$. We evaluate the proposed supervised data subset selection framework also on the same sequence of subset sizes so that it can be compared with the batch active learning results. We construct a random sampling baseline for comparison, where we randomly sample the data set at appropriate sizes for labeling. The similarity between any pair of documents is defined as the cosine similarity between their TF-IDF representations. For FASS, we fix $\beta_t = \beta = 4000, \forall t$ and test four different submodular objectives: f_{NB}, f_{NN}, f_{fac}, and f_{fs} ($c = 0.1$). In addition, we construct another baseline (FASS+RS), which is implemented the same as FASS, except that in each iteration, the submodular optimization procedure in Line 7 is replaced with a random sub-sampling strategy.

Naïve Bayes Classifier: We first explore the NB classifier under the bag-of-words model. We apply a Laplace smoothing parameter of 0.02 for training all NB models in the experiments. We evaluate the supervised data subset selection framework (SS+f_{NB})

[1] Data is obtained at http://qwone.com/~jason/20Newsgroups/



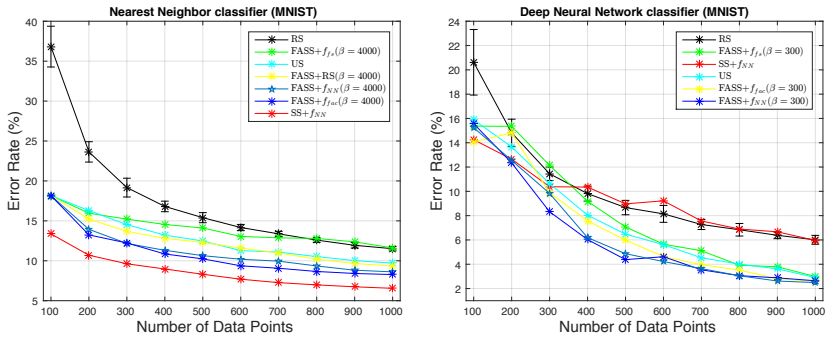
(a) (b) (c)

Figure 4.1: Text categorization: classification error evaluated on (a) NB classifier; (b) NN classifier; and (c) LR classifier for different subset sizes chosen by uncertainty sampling (US), random sampling (RS) (error bars indicate standard deviation over multiple random draws), FASS with f_{fs}, f_{fac}, f_{NB}, f_{NN}, supervised data subset selection (SS) with f_{NB} or f_{NN} (SS+$\{f_{\text{NB}}$, $f_{\text{NN}}\}$). Error rates for NB, NN, and LR classifiers trained on the whole set are 11.1%, 19.1%, and 11.7%.

as Problem 4.6 with the objective $f_{\text{NB-text}(\alpha=0.02)}$. As shown in Figure 4.1a, SS+f_{NB} and FASS+f for any choice of f perform consistently superior to random sampling (RS), which outperforms the uncertainty sampling (US) at all sizes. FASS+RS is significantly outperformed by FASS+f for any f. Drastic improvement at small sizes is achieved by SS+f_{NB}, which is outperformed by FASS+f_{NB} at larger sizes. Comparing different f in FASS+f, f_{NB} performs the best.

Nearest Neighbor Classifier: Next, we focus on how the proposed approaches perform on training NN classifiers. We evaluate the supervised data subset selection framework (SS+f_{NN}) formulated as Eqn 4.18 with f_{NN}. Unlike NB classifier, NN is not a probabilistic model. We model the posterior probability of a sample u given by a labeled training set S as

$$p(y|x_u;\theta(S)) = \frac{\exp\left(\max_{j\in S\cap V^y} w(u,j)\right)}{\sum_{y'\in\mathcal{Y}} \exp\left(\max_{j\in S\cap V^{y'}} w(u,j)\right)}. \tag{4.25}$$



(a) (b)

Figure 4.2: Handwritten digit recognition: classification error evaluated on (a) NN classifier and (b) DNN classifier for various subset sizes chosen by different methods. The error rates for NN and DNN classifiers trained on the whole set are 3.1% and 1.0%.

As shown in Figure 4.1b, SS+f_{NN} yields superior performance across the board over all other subset selection methods. The performances of different FASS schemes are ordered as $f_{\text{fac}} > f_{\text{NN}} > f_{\text{NB}} > f_{fs}$. Superior results are achieved with f_{fac} or f_{NN}, either of which matches well with the Nearest Neighbor classifier. Between f_{NN} and f_{fac}, f_{fac} always performs better, which may be due to the fact that the effectiveness of f_{NN} is very sensitive to the accuracy of the hypothesized class labels on which it is defined.

Logistic Regression Classifier: Lastly, we extend to select data for training an LR classifier, which is formulated and solved by the LIBLINEAR tools [Fan et al., 2008]. The results are shown in Figure 4.1c. Although f_{NN} and f_{NB} are not derived based on the LR model, superior results are still observed with them in the supervised setting. Between f_{NN} and f_{NB}, f_{NN} performs better, which indicates that f_{NN} may fit better with the properties of the LR classifiers. Similar to the results in the NN classifier, FASS with f_{NN} and f_{fac} perform better than other objectives, and yield performance competitive with SS+$\{f_{\text{NN}}, f_{\text{NB}}\}$ at large subset sizes.

4.3.2 Handwritten Digit Recognition Experiments:

Experimental Setup: We evaluate the handwritten digit recognition task on the MNIST database [2], which consists of 60,000 training and 10,000 test samples. Each data sample is an image of a handwritten digit. The training and test data are both almost evenly divided among 10 different classes. The goal is to classify each image as a digit. Different from the setup for the text categorization experiments, we only run one instance of each subset selection strategy except for the random sampling baseline, since the training and test data are fixed. We run 10 instances of random draw for the random sampling baseline, and report the averaged classification error as its performance. For batch active learning, we also experiment with $B = 100$, $T = 10$ and $k = 1000$. We bootstrap the batch active learning with a different strategy: instead of randomly selecting B examples, we label a set of B representative data instances by solving $\max_{|S|=B;S\subseteq\mathcal{U}} f_{\text{fac}}(S)$ (f_{fac} does not assume any labels). On this task, we examine how various subset selection strategies perform on NN and DNN classifiers. The NB classifier is not included since it does not fit with this task. We didn't evaluate SS+f_{NB} or FASS+f_{NB} for comparison either, since the proposed Naïve Bayes submodular function f_{NB} is defined on discrete features, i.e., a set features that take categorical values.

Nearest Neighbor Classifier: First, we evaluate the proposed framework on NN classifiers. The similarity between any pair of data instances i and j is measured as $d - \|x^i - x^j\|_2^2$, where $d = \max_{u,u'\in V} \|x^u - x^{u'}\|_2^2$. We represent the feature x^i of each image i as the vector of its pixel values. We compare different instances of FASS with $\beta = 4,000$. The results in Figure 4.2a show similar trends to the 20 Newsgroup experiments under the NN classifier.

Deep Neural Network Classifier: Lastly we test on DNN based classifiers. A DNN model, which consists of two convolution layers followed by two fully connected layers, is trained using Caffe [Jia et al., 2014] on the set of labeled images selected by each ap-

[2]The data set is downloaded from yann.lecun.com/exdb/mnist

proach. We report the results in Figure 4.2b. SS+f_{NN} performs well at small sizes and then matches the random baseline. This indicates that f_{NN}, though performing well on NN and LR classifiers for the supervised setting, does not fit with the properties of the DNN model. Interestingly, drastic improvements are achieved by the uncertainty sampling strategy, which suggests that manually labeling the samples that are uncertain to the current system is very valuable for updating the DNN model. Different from other classifiers, FASS+f tends to perform well when β is small and in the range $[300, 1000]$. Here we show results for $\beta = 300$. Significant improvements are achieved by FASS+f_{NN} or f_{fac}. Though formal analysis for the DNN model is not available, the empirical results suggest that it is beneficial to select a set of uncertain data instances that are representative about the whole set as well.

4.3.3 Choice of β for FASS

In this part, we discuss the interplay of the choice of β for FASS schemes and its performance. We show the comparison for NB, NN, and LR classifiers on text categorization experiments in Figure 4.3a, 4.3b, and 4.3c, respectively. Figure 4.3a and 4.3b show that FASS schemes are, in general, not sensitive to the choice of β when β ranges between $[2000, 6000]$ under NB and NN classifiers. In Figure 4.3c, we observe that the performance of FASS+f_{fac} varies as different choices of β. However, consistent and significant improvements are achieved with each choice of β for FASS+f_{fac} over the random baseline. Similarly, we observe that FASS with different submodular objectives are not sensitive to β under either NN (Figure 4.3d) or DNN (Figure 4.3e) classifier for the handwritten digit recognition task.

4.4 Discussion

As an extension to our work, we would like to look at the likelihood, or more generally risk, of a large family of classifiers as a function of subsets of training data and from the perspective of submodularity. Given the enormous empirical success of deep neural networks (DNNs), it would also be useful to complement our currently empirical-only results of Figure 4.2b that, while showing good performance, could be significantly improved with a submodular

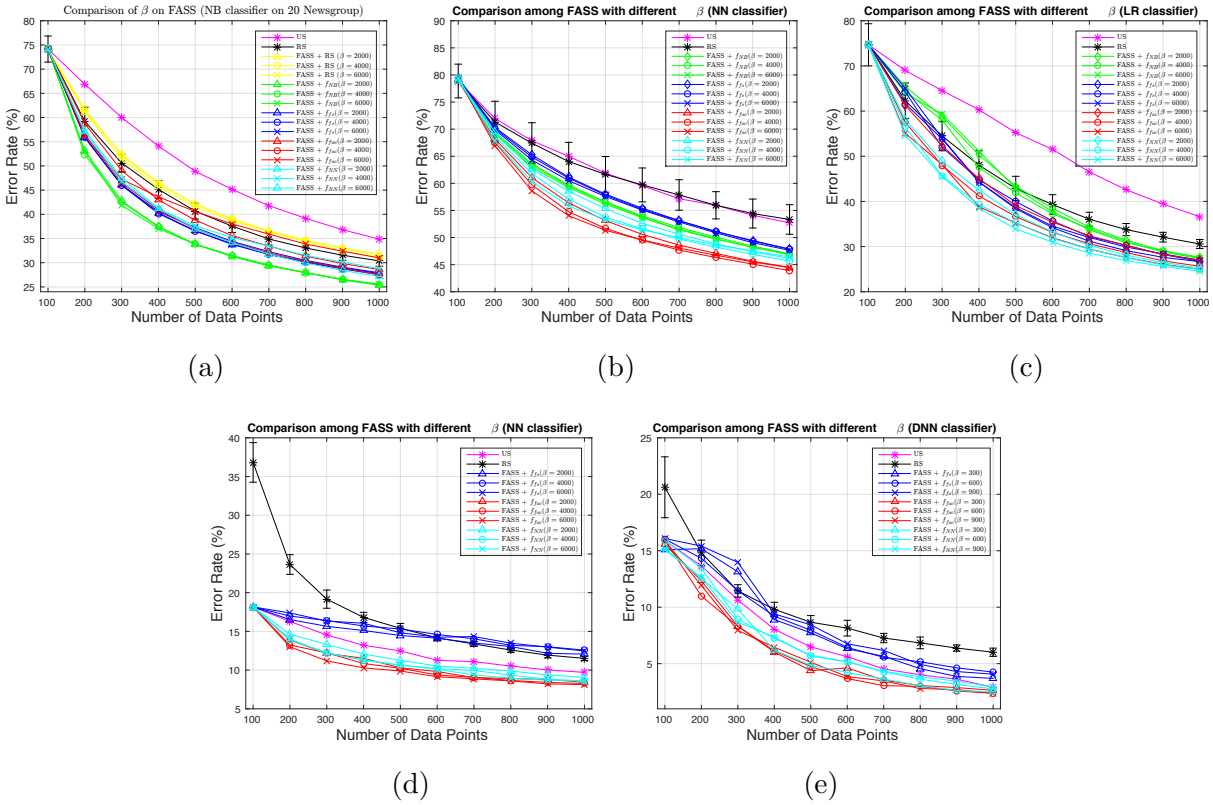


Figure 4.3: Comparison among FASS with different β on text categorization experiments (first row) and handwritten digit recognition experiments (second row).

function that better matches the properties of the DNN.

Chapter 5

INTERACTIVE LEARNING OF SUBMODULAR MIXTURES

Given the success of the submodular data summarization paradigm for the three case studies, we are now confident that submodular set functions can naturally capture the notions of representativeness, coverage, diversity, and information for a number of tasks. In some cases, there is an underlying submodular function that fully characterizes the utility of a data set as we have shown in Chapter 4. However, it is more often the case that the underlying utility function, though submodular, cannot be easily analyzed, or even non-submodular. It is still of great interest to express the utility function in terms of a submodular function so that the amenable optimization properties of submodularity can be leveraged. This is the key motivation behind our submodular data summarization framework for the case studies on the speech data subset selection and the genomics assay panel selection. In both case studies, we proposed several classes of submodular functions as the surrogate objective to approximate the underlying utility function in these tasks. These functions were chosen based on either the domain knowledge or the validation on the experiments. We have not yet, however, attempted to further improve the goodness of the submodular surrogate functions for these tasks. In other words, the surrogate utility functions are not "learned" so as to better approximate the utility function of the underlying task.

In this chapter, we move a step forward and study a principled method to learn a submodular function as the utility model. Several variants for learning submodular functions have been studied in the literature. For example, [Goemans et al., 2009] study the following problem: given an unknown monotone submodular function f, how to efficiently find a surrogate submodular function $\hat{f}$ that approximates the unknown function everywhere, i.e., $\hat{f}(A) \leq f(A) \leq g(n)\hat{f}(A), \forall A$. They give an ellipsoid-based approximation scheme for con-

structing surrogate functions that is guaranteed to bound the unknown function f everywhere with $g(n) = O(\sqrt{n}\log n)$. Moreover, they show that the problem is information theoretically hard for any polynomial time algorithm to achieve a guarantee of $O(\sqrt{\frac{n}{\log n}})$. Another variant of the learning problem was studied by [Balcan and Harvey, 2011]. They consider learning an unknown submodular in a PAC setting. Given valuations of the submodular function f on subsets sampled from a distribution, they give a learning algorithm that approximates f on subsets sampled from the same distribution with high probability and with a factor $O(\sqrt{n})$. They show that the problem has a lower bound of $O(n^{1/3})$. Therefore, approximating a submodular function everywhere (with or without a distributional assumption) is theoretically hard.

Instead, we adopt a much simpler learning paradigm that we believe is more practical in our applications. The learning paradigm is based on a given set of fixed submodular components. Formally, given the variety of information that may be captured by different submodular functions, it is often desirable to design a family of d submodular function components $\{f_i\}_{i=1}^d$ and express the utility set function for an underlying task as a non-negative weighted mixture $\sum_{i=1}^d w_i f_i$ of these fixed submodular components with $[w_1, \ldots, w_d]^\top = w \in \mathbb{R}_+^d$. Note that $\sum_{i=1}^d w_i f_i$ is also submodular. Each non-negative weight w_i represents the importance of the information expressed by the corresponding submodular function. When the number of submodular components d is small, the weights may be hand-tuned to better fit for the task. It is of much more interest and practical utility, however, to learn a weighted mixture of these submodular components in an automatic fashion, a problem we call *learning of mixtures of submodular functions* (LMSF).

The offline version of LMSF has been extensively studied for a number of machine learning tasks, including document summarization [Lin and Bilmes, 2012, Sipos et al., 2011, Bairi et al., 2015], image summarization [Tschiatschek et al., 2014], and video summarization [Gygli et al., 2015]. Given a family of fixed submodular components, this work learns a weighted mixture from a batch of training data that consists of a set of (collection,summary) pairs. Taking an image summarization task as an example, a collection is a large set of

images, and a summary is a small subset of chosen images — often by a human annotator — to best represent the collection. The approach formulates LMSF as a supervised learning problem via large-margin structured prediction, in order to optimize the weight vector w such that the human summaries scored by the learned submodular function are separated from competing summaries by a large margin. The learning framework is offline in nature, since the training data needs to be available up front, before learning begins.

Though showing great success on a number of tasks, the applicability of LMSF in the offline setting is significantly limited by the challenges of training data collection. Learning a good submodular function often relies on a large amount of high-quality training samples (ground truth (collection,summary) pairs). For most summarization tasks, producing good summaries from a large collection of items is costly and error-prone, especially when the data is acquired via a crowdsourcing marketplace. Another relevant application is redundancy removal, via summarization, in machine learning training data sets as investigated in the case studies in the speech data subset selection and the batch active learning; here evaluation of a summary involves training on a subset of the training data followed by evaluating on a test set. Learning a submodular mixture as a surrogate objective for this process would require producing training data subsets of high quality, i.e., multiple "core sets" for machine learning problems [Agarwal et al., 2005]. For many modern machine learning methods (e.g., deep models) this is currently impractical. Therefore, preparing high-quality summarization training datasets, in general, can be extremely time-consuming, taking months or years to produce. It is this problem that we address in this work.

For the problems such as the above, one viable option is to approach LMSF from an interactive setting, as we do in this work. By interactive, we mean a weighted mixture of submodular functions is continuously learned while the training samples are collected on the fly. Evaluation is always done on candidate summaries generated by the learner (i.e., a human would not be faced with the arduous task of summarizing an image collection and rather would only be faced with evaluating the quality of summaries that are much smaller; a machine learning training/testing system need not repeatedly train and test on the full

training data, but rather only quickly evaluate the quality of a relatively small training data set). To be precise, an interactive learning problem consists of two components – a learner and an evaluator. At each round $t = 1, \ldots, T$, the learner interacts with the evaluator in the following way:

1. *Query selection:* A collection of items V_t is revealed to the learner. The **learner** chooses a summary $A_t \subseteq V_t$ (e.g., a subset of fixed size k) from the collection based on the data V_t and the feedback from previous rounds.

2. *Feedback collection:* The evaluator provides feedback $y(A_t)$ on the chosen summary A_t.

3. *Parameter estimation:* The learner updates the estimate of the weight vector w.

A critical goal is to design a learner that identifies the most informative summary to query in each round so as to learn a good mixture $\sum_i w_i f_i$ using a minimum number of queries. We call this problem *interactive learning of mixtures of submodular functions* (ILMSF). In this setting, we require that the feedback $y(A_t)$ is a single score (which may take a discrete or real value depending on the application) representing the quality of the summary as a whole, and that no additional feedback is provided (e.g., no feedback on the individual items $a \in A_t$).

We are unaware of previous work that studies such interactive learning with a feedback model similar to the above, despite — as we will see — the utility of our model for a number of applications.

One line of work similar to ILMSF is [Yue and Guestrin, 2011, El-Arini et al., 2009, Raman et al., 2012]. They focus on learning a mixture of submodular components in the context of online learning, where the goal is to maximize the cumulative quality feedback scores of the chosen summaries (equivalently, minimize the cumulative regret). In this work, they consider applications such as news recommendations and web search engines, and utilize the weighted mixture of submodular components as the utility model. Our problem setting differs from such work, however, in both the learning goal and the feedback model. Our

aim is to learn a good utility function with few queries, whereas their approaches focus on choosing summaries to minimize the cumulative regret. Our feedback model assumes only a single number representing the quality of the summary A_t, whereas their work requires (cumulative) feedback for each individual item $a \in A_t$.

Other relevant work includes [Singla et al., 2016], where they consider the application of summarizing content (e.g., an image collection) by leveraging users' feedback. Assuming that the utility function is an unknown submodular function whose noisy evaluations are accessible from the users' feedback, their goal is to maximize the utility function in an interactive manner while minimizing the cost of user queries. Although related in spirit, their problem setting differs from ILMSF in both the query model and the utility model. In ILMSF, at round t, we query the quality of a complete and composed summary $A_t \subseteq V_t$, whereas the query setting in [Singla et al., 2016] focuses on obtaining the feedback on the gain of adding an item to an existing subset. An unknown and abstract submodular function is utilized in [Singla et al., 2016] to model the utility, whereas we assume the utility function admits a representation as a weighted mixture of known submodular components.

In this chapter, we propose a novel setting for interactively learning mixtures of submodular functions. We demonstrate that ILMSF naturally occurs in a number of machine learning applications, including speech data subset selection, data collection summarization using users' feedback, and personalized recommendation of media contents.

To address ILMSF, we propose a novel algorithmic framework–ActiveCombSampling. In contrast to the large-margin framework often used for LMSF, we adopt the linear regression framework for estimating the parameter w, which easily handles the varying nature of the quality feedback scores in ILMSF. For the query selection problem, we approach it from the active learning perspective. In particular, we utilize a simple uncertainty sampling strategy. The strategy is implemented as follows: in each round t, the algorithm identifies and queries the summary A_t that has the largest uncertainty (defined in Sec 5.2.2). Given the combinatorial structure of the problem, the pool of all possible summaries is exponential in the size of the collection V_t. Clearly, a brute-force search for the summary that has the

maximum uncertainty is infeasible. Instead, we formulate the search problem as combinatorial optimization, where the objective set function represents the variance for estimating the quality score of each subset. Though we demonstrate the inapproximability of this optimization problem, we show, for the applications considered in this work, that this objective can be naturally decomposed as a difference of submodular (DS) functions. Therefore, the underlying problem (Problem 5) can be efficiently optimized with the submodular-supermodular procedures that admit additive approximation guarantees [Narasimhan and Bilmes, 2005,Iyer and Bilmes, 2012, Byrnes, 2015].

Next, we show that the proposed framework ActiveCombSampling can be slightly modified to handle ILMSF under the regret minimization setting. This setting asks for a learner to choose summaries such that their cumulative quality feedback scores are maximized. A critical goal for this setting is to optimize the exploration-exploitation trade-off (Section 5.4) at the query selection step. Interestingly, we demonstrate that the trade-off can be easily modeled by a slight modification of the objective function for ActiveCombSampling.

Lastly, we empirically evaluate the proposed ActiveCombSampling for ILMSF. In particular, we demonstrate its efficacy on simulated data, and on real-world applications, including image collection summarization, and speech training data subset selection. We show that these tasks can be learned in the proposed interactive learning framework, and, moreover that, given the same number of queries, the proposed ActiveCombSampling is shown to consistently outperform the baseline methods.

5.1 Problem Formulation

In this section, we formally define the setting of ILMSF. Let T be the total number of rounds. For each round $t = 1, \ldots, T$, a collection of items V_t is made available to the learner. A family of d submodular components $\{f_i^t\}_{i=1}^d$ is also accessible to the learner. We clarify that each component function $f_i^t : 2^{V_t} \to \mathbb{R}_+$ is indexed by the round t via the ground set V_t, which may vary in t. However, the analytical definition of each component f_i^t does not change with

t. Each subset $A \subseteq V_t$ is represented by a d-dimensional vector $f_A^t = [f_1^t(A), \ldots, f_d^t(A)]^\top$. In the query selection step, the learner composes and queries a summary $A_t \subseteq V_t$ from the collection. We assume all summaries satisfy a cardinality constraint $|A_t| = k$. As shown below, our proposed algorithm, however, can also handle more general cases such as knapsack and matroid constraints.

Given the queried summary A_t, the system, next, offers feedback $y(A_t)$ to the learner, which then updates the estimate of w. We assume that the feedback $y()$ for any subset A is randomly and independently sampled from some distribution with mean $w^\top f_{A_t}^t$ and variance σ^2, where $w \in \mathbb{R}^d$ is the unknown and desired parameter. Without loss of generality, the distribution is presumed Gaussian, i.e., $\mathcal{N}(w^\top f_{A_t}^t, \sigma^2)$, since, for applications where the Gaussian assumption does not hold, one can always query A_t multiple times (e.g., query the same summary from different users) and take their sample mean as its feedback $y(A_t)$, which asymptotically is a random sample from the Gaussian distribution with mean $w^\top f_{A_t}^t$ by the central limit theorem.

The objective of ILMSF is to learn the unknown parameter w accurately using a minimum number of queries.

5.2 An Algorithmic Framework for ILMSF

In this section, we study both the parameter estimation and the query selection problem for ILMSF. In particular, we formulate the parameter estimation problem as an instance of linear regression. We then show how to approach the query selection problem using the uncertainty sampling strategy. Combining these two components together leads to the proposed algorithm–ActiveCombSampling (Alg 7).

5.2.1 Parameter Estimation of w

At the end of any round t, a set of t training samples $\mathcal{D}_t = \{f_{A_\tau}^\tau \in \mathbb{R}_+^d, y(A_\tau) \in \mathbb{R}_+\}_{\tau=1}^t$ has been collected. Given the feedback model that assumes $y(A_\tau)$ to be independently sampled from the Gaussian distribution $\mathcal{N}(w^\top f_{A_\tau}^\tau, \sigma^2)$ for any time τ, estimating the parameter w

from the training set $\mathcal{D}_t$ can be easily formulated as a linear regression problem. Applying a Gaussian prior on the parameter w as $\mathcal{N}(\mu_0, \Sigma_0)$, the posterior distribution $p(w|\mathcal{D}_t)$ can be derived in a closed form as a Gaussian distribution $\mathcal{N}(\hat{\mu}_t, \hat{C}_t)$ (detailed derivations are given in the Appendix A.2), where

$$\hat{C}_t = \Big(\Sigma_0^{-1} + \frac{1}{\sigma^2}\sum_{\tau=1}^{t} f^{\tau}_{A_\tau}(f^{\tau}_{A_\tau})^\top\Big)^{-1}; \tag{5.1}$$

$$\hat{\mu}_t = \hat{C}_t\Big(\Sigma_0^{-1}\mu_0 + \frac{1}{\sigma^2}\sum_{\tau=1}^{t} y(A_\tau) f^{\tau}_{A_\tau}\Big). \tag{5.2}$$

Observe that $\hat{\mu}_t$ is the maximum a posterior (MAP) estimate of the parameter w given the data $\mathcal{D}_t$, and the positive semidefinite matrix $\hat{C}_t$ measures the covariance for estimating w. As we will see, $\hat{C}_t$ will be used for query selection.

Instead of the large-margin formulation previously employed for the offline LMSF, we adopt the linear regression framework for estimating the parameter w from the training samples $\mathcal{D}_t$. The reason for choosing the regression framework is due to the nature of the training data available for the interactive setting. The training data for the offline LMSF often consists of a set of ground-truth (collection,summary) pairs, whereas, for ILMSF, the training sample collected at any round t can be regarded as a tuple of three elements $\{V_t, A_t, y(A_t)\}$. Unlike the handcrafted summaries for LMSF that are deemed high quality, the queried summaries for ILMSF may have quality scores that range between low and high, or that are more fine-grained as real values depending on the applications. The regression framework can easily handle this varying nature of the quality feedback scores.

5.2.2 Query Selection

Formulation: Another key problem regarding ILMSF is how to select the summary to query so as to improve the estimate of the parameter w the best. This problem can be cast as an instance of active learning for training linear regression models. Although it has been previously studied in the active learning literature [Settles, 2009, Chaudhuri

Algorithm 7: ActiveCombSampling

Input: μ_0, Σ_0, σ^2, d, T, V_t and $\{f_i^t\}_{i=1}^d$ for $t = 1, \ldots, T$.

Initialize $\hat{C}_0 = \Sigma_0$.

for $t = 1, \ldots, T$ **do**

  Define $v(A) = \sum_{i=1}^{d}\sum_{j=1}^{d} \hat{C}_{t-1}^{i,j} f_i^t(A) f_j^t(A)$

  $A_t \in \text{argmax}_{|A|=k, A\subseteq V_t} v(A)$ // Solved via DS optimization;

  $\hat{C}_t \leftarrow \left(\Sigma_0^{-1} + \frac{1}{\sigma^2}\sum_{\tau=1}^{t} f_{A_\tau}^\tau (f_{A_\tau}^\tau)^\top\right)^{-1}$// Covariance matrix;

  $\hat{\mu}_t \leftarrow \hat{C}_t\left(\Sigma_0^{-1}\mu_0 + \frac{1}{\sigma^2}\sum_{\tau=1}^{t} y(A_\tau) f_{A_\tau}^\tau\right)$ // Estimate of w;

end for

Output $\hat{w} \leftarrow \hat{\mu}_T$.

et al., 2015, Gu et al., 2012, MacKay, 1992], most existing approaches do not directly apply to the setting of ILMSF, since they, in general, require enumerating over the pool of all unlabeled data items. Let $\mathcal{A}_t$ be the set of all candidate summaries for querying in round t ($|\mathcal{A}_t|$ is exponential in $|V_t|$), the active learning approaches in [Chaudhuri et al., 2015, Gu et al., 2012] require computing a Fisher information matrix over the pool of all candidate summaries: $I_{\mathcal{A}_t} = \sum_{A\in\mathcal{A}_t} f_A^t (f_A^t)^\top$, which is clearly infeasible in this case.

To this end, we appeal to the simple and also commonly used query selection strategy: *uncertainty sampling* [Lewis and Gale, 1994]. The idea of the uncertainty sampling is to, in each round, query the most informative summary, whose informativeness is measured by the uncertainty of estimating its quality score. While having an exponential searching complexity, the uncertainty sampling method, as we will see, can be formulated as a difference of submodular (DS) optimization, which admits simple and efficient optimization algorithms [Iyer and Bilmes, 2012]. Moreover, the uncertainty sampling criterion may be slightly modified leading to a sampling strategy that handles ILMSF under the regret minimization setting (Section 5.4).

We next discuss how to apply the uncertainty sampling strategy to our query selec-

tion problem. At any given round t, the posterior distribution $p(w|\mathcal{D}_{t-1})$ has the form $\mathcal{N}(\hat{\mu}_{t-1}, \hat{C}_{t-1})$. Therefore, the distribution of the estimated quality score for any summary $A \subseteq V_t$ can be written as $p(w^\top f_A^t|\mathcal{D}_{t-1})$, which is also a Gaussian distribution $\mathcal{N}(r(A), v(A))$ with mean $r(A) = \hat{\mu}_{t-1}^\top f_A^t$ and variance $v(A) = (f_A^t)^\top \hat{C}_{t-1} f_A^t$. The uncertainty sampling strategy is to identify the summary $A \subseteq V_t$ of size k that has the largest variance $v(A)$. More formally, the uncertainty sampling problem can be formulated as a combinatorial optimization:

$$\max_{|A|=k, A\subseteq V_t} v(A), \tag{5.3}$$

where $v(A) = (f_A^t)^\top \hat{C}_{t-1} f_A^t$ with $f_A^t = [f_1^t(A), \ldots, f_d^t(A)]^\top$ for all $A \subseteq V$. Observe that $v : 2^{V_t} \to \mathbb{R}_+$ is a set function that maps any subset $A \subseteq V_t$ to a non-negative real value, since the covariance matrix $\hat{C}_{t-1}$, as defined in Eqn. (5.1), is a positive semidefinite matrix. Denoting the $(i,j)^{th}$ entry of the maxtrix $\hat{C}_{t-1}$ as $\hat{C}_{t-1}^{i,j}$, the objective set function $v(A)$ can also be decomposed as follows:

$$v(A) = \sum_{i=1}^{d} \sum_{j=1}^{d} \hat{C}_{t-1}^{i,j} f_i^t(A) f_j^t(A), \tag{5.4}$$

which is the weighted sum over the product between every pair of submodular components f_i^t and f_j^t. Note that the weights $\hat{C}_{t-1}^{i,j} \in \mathbb{R}$ may take negative values.

Hardness: Our next goal is to study the optimization problem as defined in Eqn. (5.3). Clearly, brute-force searching for the optimal summary requires exponential complexity. It would be ideal if the objective $v(A)$ can be shown as submodular, leading Eqn. (5.3) to a submodular optimization problem, which one can solve efficiently and near-optimally. Unfortunately, $v(A)$, in general, is not submodular. In fact, we show that Eqn. (5.3) is hard to optimize within any polynomial factor, even if the family of submodular components are chosen from any general submodular functions.

Theorem 11. *Unless $P = NP$, there cannot exist any polynomial time approximation algorithm for Eqn.* (5.3)*. In particular, let $n = |V_t|$ and $0 < \alpha(n) \leq 1$ be any positive*

function of n, if there exists an algorithm that approximates Eqn. (5.3) *within the factor $\alpha(n)$, then $P = NP$.*

Proof. Similar to the idea employed in the proof of Theorem 5.1 in [Iyer and Bilmes, 2012], we prove this hardness result by reducing Eqn. (5.3) to the *subset sum* problem. Given a positive modular function m and a positive constant t, the subset sum problem asks if there is any subset $S \subseteq V$ such that $m(S) = t$. First we choose a random subset C of size k (unknown to the algorithm), and define $t = m(C)$. Define a set function v such that $v(A) = \alpha(n)$ if $m(A) \neq t$ and $v(A) = 1 + \epsilon$ otherwise. Consider the optimization problem $\max_{A \subseteq V, |A|=k} v(A)$. Suppose that the problem can be approximated within the factor $\alpha(n)$ by some polynomial time algorithm, it implies that the algorithm is guaranteed to find a subset A of size k such that $v(A) \geq \alpha(n)(1 + \epsilon) > \alpha(n)$, thereby, the solution A must have value $v(A) = 1 + \epsilon$. The algorithm then solves the subset sum problem, which is a contradiction unless $P = NP$.

Next, we show that the set function $v(A)$ can be written as

$$v(A) = [f_1(A), \ldots, f_d(A)] C [f_1(A), \ldots, f_d(A)]^\top$$

for some positive semidefinite matrix C and some vector of d submodular components. Consider the instance of $d = 2$ and $C = [1, -1; -1, 1]$. Let $v'(A) = \sqrt{v(A)}$. Let $\alpha \triangleq \min_{X \subset Y \subseteq V \setminus j} v'(j|X) - v'(j|Y)$, where $\alpha \geq 2(\sqrt{\alpha(n)} - 1)$. Let $f_1'(A) = \sqrt{|A|}$ and $\beta \triangleq \min_{X \subset Y \subseteq V \setminus j} f_1(j|X) - f_1(j|Y) = 2\sqrt{n-1} - \sqrt{n-2} - \sqrt{n}$. Let $f_2(A) = v'(A) + \frac{\sqrt{\alpha}}{\beta}\sqrt{|A|}$. Let $f_1(A) = \frac{\sqrt{\alpha}}{\beta} f_1'(A)$. It is easy to verify that both f_1 and f_2 are submodular. Moreover, Using f_1 and f_2 as the submodular components, we can derive that $[f_1(A), f_2(A)] C [f_1(A), f_2(A)]^\top = (f_1(A) - f_2(A))^2 = v(A)$. Therefore, there cannot exist any polynomial time algorithm that solves Eqn. (5.3) within any factor $\alpha(n)$ unless $P = NP$.

□

Moreover, even if the family of submodular components $\{f_i^t\}_{i=1}^d$ is constrained to be as simple as modular set functions, namely, each component f_i^t satisfies $f_i^t(A) = \sum_{a \in A} f_i^t(a), \forall A \subseteq V_t$, Eqn. (5.3) is still not easy to optimize as shown below:

Theorem 12. *Suppose* $\{f_i^t\}_{i=1}^d$ *are all modular functions, there does not exist any polynomial time algorithm that approximates Eqn.* (5.3) *within any constant factor if the Random* k*-SAT hypothesis is true.*

Proof. The key idea is to reduce Eqn. (5.3) to the well-known densest-k-subgraph problem [Feige et al., 1997].

For simplicity of notations, we denote $\hat{C}_{t-1}$ as C, the $(i,j)^{th}$ entry of C as $C_{i,j}$, and each submodular component f_i^t as f_i. Using the modular assumption about each component function f_i, we have the following:

$$v(A) = \sum_{i=1}^{d}\sum_{j=1}^{d} C_{i,j} f_i(A) f_j(A) \tag{5.5}$$

$$= \sum_{i=1}^{d}\sum_{j=1}^{d} C_{i,j} \sum_{a\in A}\sum_{a'\in A} f_i(a) f_j(a') \tag{5.6}$$

$$= \sum_{a\in A}\sum_{a'\in A} \{\sum_{i=1}^{d}\sum_{j=1}^{d} C_{i,j} f_i(a) f_j(a')\} \tag{5.7}$$

$$= \sum_{a\in A}\sum_{a'\in A} s_{a,a'}, \tag{5.8}$$

where $s_{a,a'} = \sum_{i=1}^{d}\sum_{j=1}^{d} C_{i,j} f_i(a) f_j(a') = f_a^\top C f_{a'}$, with $f_a = [f_1(a), \ldots, f_d(a)]^\top$. $s_{a,a'}$ can be interpreted as the inner product between the vector f_a and $f_{a'}$ in the transformed vector space by the covariance matrix C. Note $s_{a,a'}$ is symmetric, but may take negative values. In this case, Eqn. (5.3) is then written in the following form:

$$\max_{|A|=k, A\subseteq V_t} \sum_{a\in A}\sum_{a'\in A} s_{a,a'}. \tag{5.9}$$

Note that this optimization problem generalizes the well-known Densest-k-Subgraph (DkS) problem [Feige et al., 1997] when $s_{a,a'}$ is symmetric and non-negative for any pair of $a, a' \in A$. Therefore, Eqn. (5.3) is at least as hard as DkS, which has been shown by [Alon et al., 2011] not to admit any constant factor approximation algorithms under the Random k-SAT hypothesis as defined in [Feige et al., 1997].

□

The Random k-SAT hypothesis is proposed in [Feige, 2002] and known as the average-case hardness assumption. Under this hardness assumption, Theorem 12 rules out the constant approximation algorithms for Eqn. (5.3) even when the submodular components are as simple as modular.

DS Optimization: Despite the inapproximability of Eqn. (5.3), it would still be desirable to design scalable and intuitive heuristics that work well in practice for optimizing Eqn. (5.3). To this end, we appeal to the difference of submodular (DS) optimization techniques that were introduced and successfully applied to a number of real-world tasks [Iyer and Bilmes, 2012, Bach, 2011, Narasimhan and Bilmes, 2005, Kawahara et al., 2011]. Given an objective set function $v(A)$, the idea of this approach is to first naturally decompose it as $v(A) = g(A) - h(A)$ with g and h being submodular. Given such decomposition, the objective may be optimized in an iterative manner, where, in each iteration, the underlying objective is approximated by a tight submodular surrogate. The efficient submodular optimization techniques can be applied to solve the subproblem in each iteration near-optimally. Given a number of submodular maximization algorithms that handle general constraints (e.g., knapsack [Gupta et al., 2010] and matroid constraints [Lee et al., 2010]), Eqn. (5.3) may be extended to these constraints and can still be optimized by the DS techniques. Given a DS decomposition of the objective $v(A)$, this paradigm provides a principled approach for applying the vast literature on submodular optimization to solve Eqn. (5.3). Despite the lack of approximation factor for Eqn. (5.3), DS optimization techniques guarantee that the solution is improved in successive iterations and is bounded with an additive error to the optimal solution [Iyer and Bilmes, 2012].

The remaining goal is to study how to naturally decompose the objective $v(A)$ as a DS function. Although any general set function can be written as a DS function, a non-trivial such construction may require exponential computational complexity [Iyer and Bilmes, 2012]. In Section 5.3, we demonstrate that deriving the DS decomposition for $v(A)$, however, is very easy for the applications considered in this work.

5.3 Applications & Submodular Components

In this section, we describe how to apply ILMSF to a number of machine learning tasks. For each application, we discuss the appropriate choices of submodular components and provide guidelines on how to decompose the corresponding query selection objective $v(A)$ as a DS function.

5.3.1 Speech data subset selection

Given a large speech data set for training automatic speech recognition systems (ASR), the goal of the speech data subset selection problem is to select a representative and informative subset of speech data and train a system on only the subset. Recall in Chapter 2, we model the utility set function r_1 for this task as a weighted mixture of submodular components, i.e.:

$$r_1(A) = \sum_{u \in U} w_u \phi(m_u(A)), \tag{5.10}$$

where U is a set of speech units (e.g., phonemes and words), w_u's are a set of non-negative weights, $\phi()$ is a concave function (e.g., $\phi(x) = \sqrt{x}$ or $\phi(x) = \log(1 + x)$), and $m_u(A) = \sum_{a \in A} m_u(a)$ is a non-negative score for the speech unit $u \in U$ in the set A, with $m_u(a) \geq 0$ measuring the relevance of the speech unit u to the data item a. In this work, the weights w_u's are predefined or hand-tuned according to the authors' intuitions and domain knowledge. As we show below, ILMSF naturally addresses the weight tuning problem in an automatic fashion. In this context, V_t is the entire speech training data and is the same in every round. At round t, a data subset A_t is chosen to train an ASR system. The quality feedback $y(A_t)$ on the subset A_t is given by the performance (recognition accuracy) of the trained system on a test set. Since training an ASR system, even on a subset of data, takes a long time, it is desirable to reduce the number of queries (and the size of the queries) needed for tuning a good weighted mixture, a goal addressed by ILMSF.

DS decomposition: By examining the definition of $v(A)$ in Eqn. (5.4), it suffices to derive $v(A)$ as a DS function, if each product term $f_i(A)f_j(A)$ is represented as a DS function. In this application, each component function admits the form: $f_i(A) = \phi(m_i(A))$. We consider three classes of commonly used concave functions:

1. $\phi_{\text{exp}}(x) = 1 - \alpha^{-x}$ with $\alpha > 1$,

2. $\phi_{\text{poly}}(x) = x^{\beta}$ with $0 < \beta \leq 1$,

3. $\phi_{\log}(x) = \log(1 + \gamma x)$ with $\gamma > 0$.

When the concave function is ϕ_{exp}, it is straightforward to derive the corresponding product as $f_i(A)f_j(A) = \phi_{\text{exp}}(m_i(A)) + \phi_{\text{exp}}(m_j(A)) - \phi_{\text{exp}}(m_i(A) + m_j(A))$, which admits the DS representation. When the concave function is ϕ_{poly}, we show below that, under mild assumptions on the data, the corresponding product is either monotone submodular or supermodular depending on β. For either case, the objective $v(A)$ becomes a DS function.

Lemma 1. *Let* $f_i(A) = \big(m_i(A)\big)^{\beta}$ *with* $0 < \beta \leq 1$ *for* $i = 1, \ldots, d$. *Given* i *and* j, *let* $r_{i,j}(a) = \frac{m_i(a)}{m_j(a)}$ *(assuming* $m_i(a), m_j(a) > 0, \forall a \in V$*). When* $0 < \beta < 1/2$, $f_i(A)f_j(A)$ *is guaranteed to be monotone submodular, if* $r_{i,j}(a) \in [\theta(\beta), 1/\theta(\beta)], \forall a \in V$, *where* $\theta(\beta) = -\frac{\sqrt{1-2\beta}}{\beta} - 1 + \frac{1}{\beta} \in (0, 1)$. *When* $1/2 < \beta \leq 1$, $f_i(A)f_j(A)$ *is guaranteed to be monotone supermodular if* $r_{i,j}(a) \in [\theta'(\beta), 1/\theta'(\beta)], \forall a \in V$, *where* $\theta'(\beta) = \sqrt{\frac{\beta - \sqrt{2\beta - 1}}{1 - \beta}} \in (0, 1)$.

The proof is given in Appendix A.3. Assuming that the ratio of the modular singleton scores $r_{i,j}(a)$ is close to 1 for all $a \in V$, the product $f_i(A)f_j(A)$ becomes monotone submodular if $\beta < 1/2$ and monotone supermodular otherwise. When $0 < \beta < 1/2$, observe that the condition $\theta(\beta)$ monotonically decreases as β decrease and $\lim_{\beta \to 0} \theta(\beta) = 0$, indicating that weaker assumption is needed as β decreases. As a concrete example, if $\beta = 0.1$, we have $\theta(\beta) \approx 0.056$ leading to the condition as $r_{i,j}(a) \in [0.056, 17.9]$, which is a reasonable assumption for real-world problems. On the contrary, when $1/2 < \beta < 1$, $\theta'(\beta)$ is a monotone decreasing function in β, suggesting that less stringent constraint is required for guaranteeing

the supermodularity as β increases. If $\beta = 0.9$, we have $\theta'(\beta) = 0.236$ leading to the condition as $r_{i,j}(a) \in [0.236, 2.806]$. Interestingly, Lemma 1 provides a full characterization of submodularity/supermodularity for $f_i(A)f_j(A)$ in terms of β, which may be of independent interests, especially, to problems that involve a product of two submodular functions of this form.

Lastly, when the concave function is $\phi_{\log}$, the product function $f_i(A)f_j(A)$ can also be shown to be submodular under conditions similar to Lemma 1.

Lemma 2. *Given i, j, $\gamma > 0$ and $\delta > \frac{e-1}{\gamma}$, let $f_i(A) = \log(1 + \gamma m_i(A))$ for $i = 1, \ldots, d$, and $r_{i,j}(a) = \frac{m_i(a)}{m_j(a)}, \forall a \in V$. $f_i(A)f_j(A)$ is guaranteed to be monotone submodular if $m_i(a) \geq \delta, \forall i = 1, \ldots, d, a \in V$ and $r_{i,j}(a) \in [\theta(\gamma, \delta), 1/\theta(\gamma, \delta)], \forall a \in V$, where $\theta(\gamma, \delta) = \log(1 + \gamma\delta) - \sqrt{\log^2(1 + \gamma\delta) - 1}$.*

Proof is in Appendix A.4. As a concrete example, if $\gamma = 10$ and $\delta = 1$, we then have that $\theta(\gamma, \delta) = 0.21$, leading to the condition of the ratio as $r_{i,j}(a) \in [0.22, 4.58], \forall a \in V$. This assumption can also be satisfied by most of the real-world applications.

5.3.2 High quality summaries generation

Recently proposed in [Singla et al., 2016], we are given one collection of data items (e.g., an image collection) and the goal is to generate a high quality summary of the collection by leveraging users' feedback. Take the image collection summarization as an example, one can define the utility function in the same form as Eqn. (5.10) via a set U of visual words (e.g., super-pixels and/or high-level concepts). The goal is to learn the weights w_u's from the users' feedback so that each learned weight w_u appropriately captures the importance of the corresponding visual word u as desired by a high quality summary. Using such utility model, one may apply ILMSF to this application as follows: in each round t, the learner first presents an image collection summary $A_t \subseteq V_t$ to a human annotator, who returns the quality feedback $y(A_t)$, and then the learner updates the estimate of the parameter w given the feedback. In this case, V_t is defined as the same image collection in every round t, and

the feedback $y(A_t)$ may be in a binary form of "like" v.s. dislike", or a more fine-grained form such as an integer scale of 1-5.

DS decomposition: Given the same utility model, the DS decomposition of the query selection objective $v(A)$ simply carries over from the speech data subset selection task.

5.3.3 Learning a user's preference model

It is often of interest to learn a preference model of a user for personalized recommendation of media contents (e.g., news articles, movies, and advertisements). This problem has been studied in a number of fields, such as diversified retrieval and online learning [Yue and Guestrin, 2011, Qin et al., 2014, Radlinski et al., 2008, El-Arini et al., 2009]. In particular, [Yue and Guestrin, 2011] and [El-Arini et al., 2009] show that a user's preference can be modeled by the class of probabilistic coverage submodular functions. Take the news recommendation as an example, the utility function is defined as:

$$r_2(A) = \sum_{u \in U} w_u \big(1 - \prod_{a \in A} (1 - p_u(a))\big), \tag{5.11}$$

where U is a set of "topics" that the user may be interested in (e.g., politics, sports, and weather), $0 \leq p_u(a) \leq 1$ is an independent probability of the article a covering topic u, and each weight $w_u \in \mathbb{R}_+$ is an unknown parameter representing the user's preference or interest level for the topic u — each component function $\big(1 - \prod_{a \in A}(1 - p_u(a))\big)$ is submodular. If the goal is to learn the user's personal preference via the weights w_u's with as few queries, this problem then naturally fits in the framework of ILMSF.

DS decomposition: In this application, the corresponding product term $f_i(A)f_j(A)$ as shown below is also easily decomposable as a DS function, leading to a DS representation of $v(A)$.

Lemma 3. *Let $f_i(A) = 1 - \prod_{a \in A}(1 - p_i(a))$ for $i = 1, \ldots, d$. Given any i and j, it holds that $f_i(A)f_j(A) = g(A) - h(A)$, where both $g(A) = f_i(A) + f_j(A)$ and $h(A) = 1 - \prod_{a \in A} \big(1 -$*

Algorithm 8: OnlineCombSampling

Input: λ, d, T, V_t, α_t, and $\{f_i^t\}_{i=1}^d$ for $t = 1, \ldots, T$.
Initialize $\hat{C}_0 = \lambda I_d$ and $\hat{\mu}_0 = 0$.
for $t = 1, \ldots, T$ **do**
    Define $v(A) = \sum_{i=1}^{d}\sum_{j=1}^{d} \hat{C}_{t-1}^{i,j} f_i^t(A) f_j^t(A)$
    Define $u(A) = \sum_{i=1}^{d} \hat{\mu}_{t-1}^i f_i^t(A)$
    $A_t \in \operatorname{argmax}_{|A|=k, A\subseteq V_t} u(A) + \alpha_t \sqrt{v(A)}$ // Solved via DS optimization;
    $\hat{C}_t \leftarrow \left(1/\lambda I_d + \sum_{\tau=1}^{t} f_{A_\tau}^\tau (f_{A_\tau}^\tau)^\top\right)^{-1}$// Covariance matrix;
    $\hat{\mu}_t \leftarrow \hat{C}_t\Big(\sum_{\tau=1}^{t} y(A_\tau) f_{A_\tau}^\tau\Big)$ // Estimate of w;
end for

$p_i(a) - p_j(a) + p_i(a)p_j(a)\big)$ *are monotone submodular.*

We give the proof in Appendix A.5.

5.4 Extension to Regret Minimization Setting

Instead of aiming to learn w with the minimum number of queries as ILMSF, an interesting variant of the problem may require a learner to choose a summary A_t in each round so as to minimize the cumulative regret:

$$\text{Regret}(T) = \sum_{t=1}^{T} \big(w^\top f_{A_t^*}^t - w^\top f_{A_t}^t\big), \tag{5.12}$$

where $A_t^* \in \operatorname{argmax}_{|A|=k, A\subseteq V_t} \sum_{i=1}^{d} w_i f_i^t(A)$. We call this scenario the *regret minimization* setting. This setting naturally occurs in a number of recommendation applications, where the goal is to recommend summaries A_t's that are deemed high quality while the unknown parameter w is learned on the fly. For the regret minimization setting, a critical goal is to optimize the exploration-exploitation trade-off at the query selection step. Exploration refers to the goal of learning w as quickly as possible, and the idea of exploitation is to

utilize the currently learned parameter w to choose a high quality summary. Interestingly, ActiveCombSampling can be slightly modified to handle the regret minimization setting. In particular, the query selection objective may be slightly modified to model the exploration-exploitation trade-off. Rather than solving Eqn. (5.3) as in ActiveCombSampling, we may solve a different combinatorial optimization problem:

$$\max_{|A|=k, A\subseteq V_t} u(A) + \alpha_t \sqrt{v(A)}, \tag{5.13}$$

where $u(A) = \sum_{i=1}^{d} \hat{\mu}_{t-1}^{i} f_i^t(A)$ is the estimated utility of A, and $v(A)$ (defined in Eqn. (5.4)) measures the uncertainty of the estimate. The parameter $\alpha_t > 0$ is a predefined function of t that controls the exploration-exploitation trade-off. We call the modified scheme OnlineCombSampling (see Alg 8). Since Eqn. (5.13) generalizes the formulation in Eqn. (5.3), the inapproximable results easily carry over to this problem. Even with the DS decomposition of $v(A)$, it is not straightforward to represent the objective $u(A) + \alpha_t \sqrt{v(A)}$ as a DS function. However, we show below that Eqn. (5.13) can be approximately transformed to the following problem:

$$\max_{|A|=k, A\subseteq V_t} u^2(A) + \alpha_t^2 v(A). \tag{5.14}$$

Lemma 4. *Suppose that $\hat{\mu}_t \geq 0$ and that an algorithm always outputs a solution $\hat{A}$ for Eqn. (5.14) with a λ-approximation ($\lambda \leq 1$), the solution $\hat{A}$ also yields a $\sqrt{\frac{\lambda}{2}}$-approximation for Eqn. (5.13).*

Proof. Since $\hat{\mu}_t \geq 0$, $u(A) = \sum_{i=1}^{d} \hat{\mu}_t^i f_i(A) \geq 0$ holds for all $A \subseteq V$. Then, we have the following:

$$(u(A) + \alpha_t \sqrt{v(A)})^2 \geq u^2(A) + \alpha_t^2 v(A) \tag{5.15}$$

$$\geq \frac{(u(A) + \alpha_t \sqrt{v(A)})^2}{2}. \tag{5.16}$$

Let A^* be the optimal solution Eqn. (5.13). Given an algorithm that always outputs a solution $\hat{A}$ for Eqn. (5.14) with λ guarantee, the following holds:

$$(u(\hat{A}) + \alpha_t \sqrt{v(\hat{A})})^2 \geq u^2(\hat{A}) + \alpha_t^2 v(\hat{A}) \tag{5.17}$$

$$\geq \lambda u^2(A^*) + \alpha_t^2 v(A^*) \tag{5.18}$$

$$\geq \frac{\lambda}{2}\{u(A^*) + \alpha_t\sqrt{v(A^*)}\}^2. \tag{5.19}$$

Therefore, it holds that $u(\hat{A}) + \alpha_t\sqrt{v(\hat{A})} \geq \sqrt{\frac{\lambda}{2}}\big(u(A^*) + \alpha_t\sqrt{v(A^*)}\big)$. □

Given the choices of the submodular components discussed in Section 5.3, the objective $u^2(A) + \alpha_t v(A)$ admits the DS representation, leading Eqn. (5.14) to a DS optimization problem. Moreover, Lemma 4 shows that the optimality guarantee of any algorithm for Eqn. (5.14) almost carries over to Eqn. (5.13).

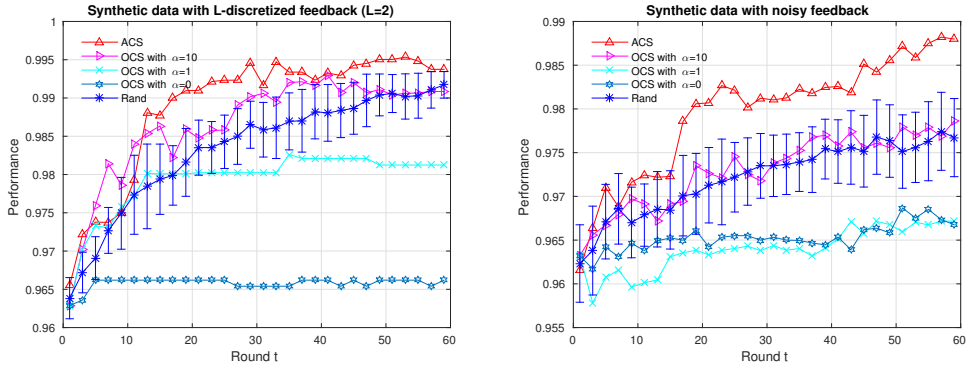
We point out the optimization objective for OnlineCombSampling (Eqn. (5.13)) matches the criterion adopted in the LinUCB algorithm, which was proposed for the contextual linear bandit problem [Li et al., 2010, Yue and Guestrin, 2011]. Assuming that Eqn. (5.13) can be solved optimally, the sublinear regret analysis for LinUCB directly applies and OnlineCombSampling is guaranteed to incur regret that grows as $O(d\sqrt{T}\log T)$.

5.5 Experimental Results

In this section, we empirically evaluate the proposed algorithm ActiveCombSampling for ILMSF on both the simulated data and the real-world applications, including the image collection summarization and the speech data subset selection.

5.5.1 Simulations on Synthetic Data

We define the utility function as $r(A) = \sum_{i=1}^{d} w_i\phi(m_i(A))$, where $\phi(x) = \log(1+x)$, $m_i(A) = \sum_{a\in A} m_i(a)$, $d = 20$, $|V| = 30$, and the unknown parameters w_i's are independently and uniformly sampled from $[0, 1]$. Each modular score $m_i(a)$ is also independently and uniformly sampled from $[0, 1]$. We evaluate the ILMSF setting with $T = 60$ and $k = 5$. We test two different quality feedback scenarios: (1) noisy feedback, (2) L-discretized feedback. In the noisy feedback setting, we simulate the feedback score for the queried summary A_t as $y(A_t) = r(A_t) + \sigma$, where σ is independently and randomly sampled from $N(0, 0.3)$. In the case of



(a) (b)

Figure 5.1: Comparison among different learning strategies: ActiveCombSampling (ACS), OnlineCombSampling (OCS) with various choices of α: 0, 1, 10, random sampling (Rand) (error bars indicate the standard deviation over 10 random instances).

L-discretized feedback, the quality feedback $y(A_t)$ is obtained by discretizing $r(A_t)$ into L bins. To be more precise, let $r_{\max} \in \operatorname{argmax}_{|A|\subseteq V_t, |A|=k} r(A)$ and $r_{\min} \in \operatorname{argmin}_{|A|\subseteq V_t, |A|=k} r(A)$ be the best and the worst score of a summary. We simulate the L-discretized feedback as:

$$y(A_t) = \lfloor \frac{r(A_t) - r_{\min}}{r_{\max} - r_{\min}} L \rfloor. \tag{5.20}$$

Therefore, the L-discretized feedback score is always an integer between 0 and L. Since $r_{\max}$ and $r_{\min}$ are NP-hard to obtain, we approximate $r_{\max}$ by running the greedy algorithm on the objective $r(A)$ and approximate $r_{\min}$ by taking the minimum score out of 200 randomly chosen summaries.

For fair comparisons, we use the same linear regression formulation for parameter estimation in each round. We compare among different query selection strategies. We construct a random sampling baseline, where, in each round, a subset of size k is randomly chosen for querying. We run 10 instances of the random sampling baseline, and report its averaged performance as well as the standard deviation. As a comparative study, we also evaluate OnlineCombSampling with various choices of α_t (cf. Eqn. (5.13)), even though we pro-

pose this algorithm for the regret minimization setting. In the experiments, we set the hyperparameter $\alpha_t = \alpha, \forall t$ and test three choices of α: 0, 1, 10.

We assess the performance of a method through the quality of the learned parameter. Let w^t be the estimated parameter at round t. Let A^* be the summary obtained by greedily optimizing $r(A)$, and A_{w^t} be the summary selected by greedily optimizing the estimated objective: $\hat{r}_{w^t}(A) = \sum_{i=1}^{d} w_i^t \phi(m_i(A))$. We report the performance as the normalized score:

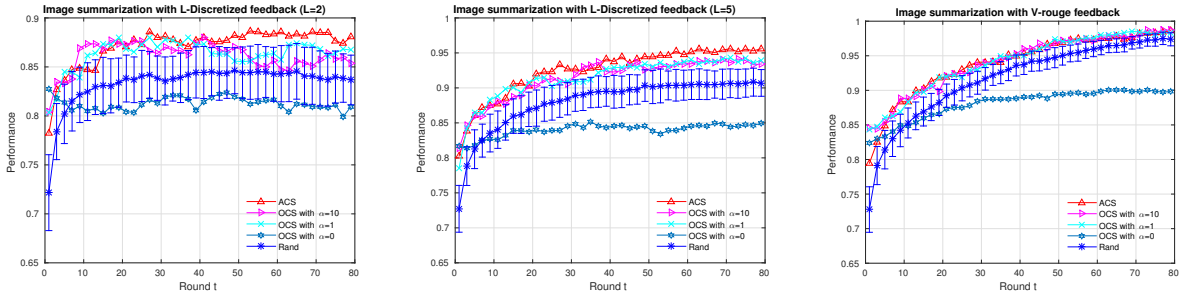
$$\bar{r}_t = \frac{r(A_{w^t})}{r(A^*)}. \tag{5.21}$$

In general, $\bar{r}_t$ is always bounded within $[0, 1]$.

The results for the 2-discretized feedback and the noisy feedback settings are shown in Figure 5.1a and 5.1b, respectively. For both feedback settings, ActiveCombSampling achieves superior results over the random baseline as well as OnlineCombSampling with all choices of α. For OnlineCombSampling, we observe that the performance gets worse as α decreases from 10 to 0, suggesting that more exploitation in the objective (smaller value of α) leads to worse performance in this task. For all methods, the performance consistently improves as the interactive learning procedure proceeds. We also note that the learned performance by ActiveCombSampling converges to about 99% of the best possible score, indicating that the learned parameter $\hat{w}$ almost converges to the unknown parameter w.

5.5.2 Image Collection Summarization

We also evaluate our framework on the image collection summarization task. We test on the image collection database created by [Tschiatschek et al., 2014]. The data set consists of 14 image collections, each comprising 100 images. For each collection, human-generated summaries are available with each summary consisting of 10 images. In this experiment, we evaluate ILMSF on each image collection separately and report the averaged results over all 14 collections. Following the evaluation criterion proposed in [Tschiatschek et al., 2014], we use the V-rouge function $r_{\text{V-rouge}}(A)$ as the unknown utility function for measuring the quality of a summary A. Given an image collection and the corresponding set $\mathcal{S}$ of human



(a) (b) (c)

Figure 5.2: Comparison for the image collection summarization task is evaluated under three different feedback settings: (1) 2-discretized V-rouge, (2) 5-discretized V-rouge, and (3) V-rouge feedback.

generated summaries, the V-rouge function is defined as:

$$r_{\text{V-rouge}}(A) = \frac{\sum_{u\in U}\sum_{S\in\mathcal{S}}\min\{m_u(A), m_u(S)\}}{\sum_{u\in U}\sum_{S\in\mathcal{S}} m_u(S)},$$

where U is a set of visual words (e.g., super-pixels or concepts), and $m_u(A) = \sum_{a\in A} m_u(a)$ is the modular function measuring the occurrence of the visual word u in the set A.

For each collection, we test ILMSF with $T = 80$. We assume that the set $\mathcal{S}$ of the human-generated summaries is not available to the learning algorithm, however, in each round t, the information about the V-rouge score of the queried summary A_t is returned to the learner. We simulate with two forms of quality feedback: (1) V-rouge feedback, (2) L-discretized V-rouge feedback. For the first case, we return $y(A_t) = r_{\text{V-rouge}}(A_t)$ as the quality feedback, whereas, in the second case, the learner receives an L-discretized V-rouge feedback, which is obtained by discretizing the V-rouge score $r_{\text{V-rouge}}(A_t)$ into L bins as defined in Eqn. (5.20).

We model the utility function via a family of submodular components $\{\phi(m_u(A))\}_{u\in U}$ with $\phi(x) = \log(1 + 100x)$ and U being the same set of visual words for defining $r_{\text{V-rouge}}$. Same as the synthetic data simulation, we report the normalized score (Eqn. (5.21)) as the performance measure. The results for 2-discretized, 5-discretized, and V-rouge quality

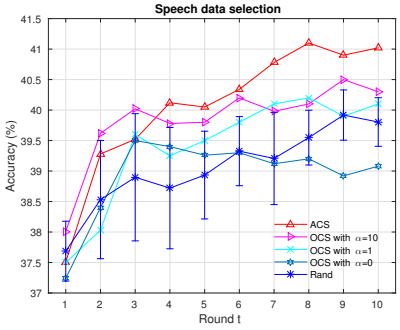


Figure 5.3: Comparison for the speech data subset selection task.

feedback settings are given in Figure 5.2a, 5.2b, and 5.2c. We observe that ActiveCombSampling yields consistent improvements over the other methods for both the 2-discretized and 5-discretized feedback settings. In the V-rouge feedback case, comparable performance is achieved by ActiveCombSampling and OnlineCombSampling with $\alpha = 1, 10$. All these methods significantly outperform the random sampling baseline. Among the three quality feedback settings, we observe that the converged performance of the V-rouge feedback case is better than the 5-discretized case, both of which outperform the 2-discretized feedback scenario. It suggests that better performance can be achieved if a more fine-grained form of quality feedback setting is adopted.

5.5.3 Speech Data Subset Selection

In the last set of experiments, we examine the task of speech data subset selection on the TIMIT data set. Given the long experimental turnaround, we test ILMSF with $T = 10$. In each round, 1% of the speech data is chosen as the training data, from which a phone recognizer is trained. The accuracy of the trained system on the test set is then returned as the quality feedback to the learner. Unlike the image collection summarization and the synthetic data simulation experiments, the utility function, in this case, cannot be represented in a closed-form and can only be evaluated via training a speech recognizer on the given subset.

We model the utility function as $\hat{r}_w(A) = \sum_{u \in U} w_u \phi(m_u(A))$, where $\phi(x) = \log(1 + 0.1x)$, U is a set of phonemes, and $m_u(A)$ is a modular function that counts the number of occurrences of the phoneme u in the subset A. Given a learning method, let w^t be the learned parameter at round t, and A_{w^t} be the data subset selected by greedily optimizing the estimated objective $\hat{r}_{w^t}(A)$. We report the performance of the learning method as the accuracy of the phone recognizer trained on the data subset A_{w^t}. We show the results in Figure 5.3. For comparison, we also report the averaged accuracy for randomly choosing 1% of the data as 38.13% with standard deviation 1.76%. We observe that the results in Figure 5.3 share the similar trend as the image collection summarization and the synthetic data simulation. In particular, ActiveCombSampling learns faster than the other methods and converges to better results.

5.6 Discussion

In this Chapter, we introduce a novel interactive setting for learning mixtures of submodular functions–ILMSF. Different from prior investigations on the similar problems, our interactive learning setting only assumes a single quality feedback score of the queried summary in each round. Such restricted quality feedback setting is more realistic to obtain the applications considered in this work, such as the image collection summarization and the speech data subset selection. For all the applications investigated here, our proposed framework leads to better empirical performance than the baseline methods. Future work will concentrate on analyzing the sample complexity of ActiveCombSampling for ILMSF. For the problem under the regret minimization setting, another extension is to analyze the sublinear convergence of the cumulative regret for OnlineCombSampling assuming that Eqn. (5.13) may be solved near-optimally.

Chapter 6

FAST MULTI-STAGE SUBMODULAR MAXIMIZATION

So far, we have mainly focused on the modeling component of the submodular data summarization paradigm for a number of machine learning tasks. In particular, we have shown that submodular set functions are often the natural model for information, representativeness, and diversity for the data summarization tasks considered thus far. Another important component regarding this paradigm is the efficiency of the optimization algorithms. Recall that the data summarization problems are often formulated as a combinatorial optimization problem as follows:

$$\max_{|S|\in\mathcal{C}} f(S), \tag{6.1}$$

where f is the submodular utility function and $\mathcal{C}$ defines the set of feasible sets as a summary. A simple form of the feasibility set is defined by the cardinality constraint leading to the following problem:

$$\max_{|S|\leq\ell} f(S), \tag{6.2}$$

where ℓ is the size constraint. Note this formulation was used for genomics assay panel selection and some of its extensions (knapsack constraint or matroid constraint) were applied to speech data subset selection and batch active learning problems. For simplicity of discussion, we focus on Problem 6.2 here, but as we will see, our study easily generalizes to other constraints as well. As discussed in Section 1.5.5, the accelerated variant of the greedy algorithm (LazyGreed) is guaranteed to near-optimally solve this problem with almost linear number of function valuations in practice.

In many cases, however, the very advanced machine learning algorithms that we need to use to process large data sources are too computationally costly for the amount of data that

exists. For example, when the data source is very large (e.g., n in the billions or trillions), even LazyGreed becomes untenable. Moreover, even smaller sized n can be prohibitive, particularly when evaluating the function f itself is expensive. For example, in document summarization [Lin and Bilmes, 2011] and speech data subset selection the graph-based submodular functions (e.g., facility location function and saturated coverage function) are often defined via a pair-wise similarity graph, having a time and memory complexity of $O(n^2)$. This is infeasible even for medium-scale values of n. Another application is the feature selection [Krause and Guestrin, 2005a,Liu et al., 2013], where a common objective is to maximize the mutual information between a given set of random variables X_A and a class C (i.e $I(X_A; C)$). The mutual information depends on computing the entropy $H(X_A)$ which can be expensive (or even exponential cost) to evaluate. Similarly, the recent promising work on determinantal point processes (DPPs) [Kulesza and Taskar, 2012], where one wants to maximize the submodular function $f(A) = \log\det(S_A)$ for a given matrix S, becomes problematic since computing log-determinant can require an $O(n^3)$ computation which is impractical already on medium-sized data sets. We therefore ask the question: are there other approaches that can address Problems (6.2) and that are scalable to very large data sets, and that still offer theoretical guarantees?

In fact, there has been great advances in addressing the sacalability issue of Problem 6.2. Existing work addresses the problem from one of the following directions:

1. streaming setting: reduce the centralized greedy algorithm to a more time and memory efficient streaming algorithm.

2. distributive setting: reduce the computationally difficult problem into many feasible subproblems that can be solved in parallel.

3. single compute setting: reduce the complexity in both the time and memory of the algorithm so that it can run feasibly on a single machine.

A detailed survey on existing work for this problem was given in Section 1.5.5. In this chapter,

we propose a multi-stage framework (MultGreed) that scale up the greedy algorithm in a single computing setting. In particular, the proposed framework directly addresses the time and memory complexity issues of running LazyGreed in three ways:

1. reducing the number of function evaluations required for the algorithm,

2. decreasing the complexity of function evaluations by using simpler surrogate (proxy) functions,

3. reducing the ground set size.

Though quite different in spirit from the distributive framework, our approach could easily be performed in concert with existing distributed algorithms. For instance, we can apply MultGreed instead of LazyGreed for solving each sub-problem in [Mirzasoleiman et al., 2013]. Conversely, their distributed procedure could also be used to solve the sub-problem in each stage of MultGreed. The theoretical analysis for both frameworks could easily be combined with each other, and they could be integrated to provide still more efficient large-scale algorithmic frameworks for these problems. Hence, our approach is *complementary* to the existing distributive architectures, although in this chapter we will concentrate on our novel multi-stage uni-processor approach.

We structure the remaining chapter as follows. Section 6.1 gives an overview of our framework. In Section 6.2, we theoretically analyze its performance while Section 6.3 offers several choices of surrogate functions for certain practical and useful classes of submodular functions. In Section 6.4, we focus on the design of MultGreed on a broad range of submodular functions, and instantiate our general framework for several submodular functions, thereby providing recipes for many real-world problems. In Section 6.5, we empirically demonstrate the performance of MultGreed.

6.1 Multi-Stage Algorithmic Framework

Often in applications, there is a desirable in quality but prohibitive in computational complexity submodular function that we shall refer to as the *target function* f. We assume a ground set size of $n = |V|$, a cardinality constraint of ℓ, and that the optimal solution to Problem (6.2) is S^{OPT}.

For completeness, we describe here how LazyGreed accelerates the naive greedy implementation. The key insight is that the marginal gain of any element $v \in V$ is non-increasing during the greedy algorithm (a consequence of the submodularity of f). Instead of recomputing $f(v|S_{i-1})$ for each v, the accelerated greedy algorithm maintains a list of upper bounds $\rho(v)$ on each item's current marginal gain. They are initialized as $\rho(v) \leftarrow f(v)$ for each $v \in V$, and sorted in decreasing order (implemented as a priority queue). In iteration i, the algorithm pops the element v off the top of the priority queue and updates the bound $\rho(v) \leftarrow f(v|S_i)$. v is selected if $\rho(v) \geq \rho(u)$, where u is at the current top of the priority queue, since submodularity in such case guarantees that v provides the maximal marginal gain. Otherwise, we appropriately place the updated $\rho(v)$ back in the priority queue and repeat.

To this end, we consider three schemes to further accelerate LazyGreed: (a) reduce the number of function evaluations (Approximate greedy), (b) reduce the complexity of function evaluations (using simpler proxy functions), (c) reduce the ground set size (Pruning). This ultimately leads to our multi-stage greedy framework MultGreed.

Approximate greedy

In this part, we introduce a mechanism called ApproxGreed, to reduce the number of function evaluations in LazyGreed. We give a theoretical analysis for ApproxGreed in Section 6.2 — the current section defines and then offers intuition for the method. The key idea of ApproxGreed is that it does not insist on finding the item that attains exactly the maximum marginal gain in each iteration, but instead, looks for an item whose marginal

gain is close to this maximum. ApproxGreed only modifies LazyGreed by weakening the selection criteria in each iteration. More formally, if an item v is selected by LazyGreed, the optimality of its marginal gain is guaranteed if the *exact condition* $\rho(v) \geq \rho(u)$ (u is the current top of the priority queue) is met. ApproxGreed relaxes this to an *approximate condition* $\rho(v) \geq \beta\rho(u)$, where $0 < \beta < 1$. Since a potentially large number of items' marginal gains need to be reevaluated until the exact condition is met, using the approximate condition could effectively reduce the number of function evaluations at a loss of the original guarantee. The parameter β controls the level of sub-optimality: the smaller β is, the number of function evaluations reduces as does the performance guarantee. In other words, ApproxGreed, as an approximate scheme to LazyGreed, has its performance guarantee carried over from that of LazyGreed, with an additional level of approximation governed by the value β (the formal guarantee of $(1 - e^{-\beta})$ is given in Lemma 6). We would like to point out the resemblance of ApproxGreed to the recently proposed fast greedy algorithm for Problem 6.2 [Badanidiyuru and Vondrák, 2014]. Similar to ApproxGreed, they seek to identify an item whose marginal gain is within a fraction β of the maximum marginal gain in each iteration and yield the an approximation factor of $(1 - e^{-\beta})$. Unlike their algorithm, ApproxGreed builds on top of the LazyGreed, hence, further exploits the submodularity. Though quite similar in spirit, ApproxGreed might run significantly faster, in practice, than their algorithm, while yielding the same performance guarantee.

ApproxGreed can be further generalized by setting the value of β individually for each iteration, i.e., a sequence $\{\beta_i\}_{i=1}^{\ell} = \{\beta_1, \ldots, \beta_\ell\}$. Intuitively, we would design $\{\beta_i\}_{i=1}^{\ell}$ to be non-decreasing, i.e., the allowed sub-optimality decreases as the algorithm proceeds. The reason for this is that a less accurate selection at the begin has a chance of being corrected by more accurate selection in later iterations. One possible schedule would be $\beta_i = c + \frac{1-c}{\ell}(i-1)$, where $c < 1$ determines the initial sub-optimality degree of the algorithm. Then, β_i grows linearly in i from c to 1, and the choice of c determines the trade-off between the running time reduction and performance guarantee loss. Given f, ℓ, and $\{\beta_i\}_{i=1}^{\ell}$, we shall instantiate the approximate greedy procedure as $S \in \text{ApproxGreed}(f, \ell, \{\beta_i\}_{i=1}^{\ell})$.

Algorithm 9: MULTGREED: A Multi-Stage Greedy Alg.

Input: f, ℓ, J, $\{f_j\}_{j=1}^{J}$, $\{\ell_j\}_{j=1}^{J}$, $\{\beta_i\}_{i=1}^{\ell}$;
Initialization: $C \leftarrow \emptyset$, $L \leftarrow 0$;
for $j = 1 \dots J$ **do**
    Define $F_j(S) \triangleq f_j(S|C)$ for all $S \subseteq V$;
    $S \in \text{APPROXGREED}(F_j, \ell_j, \{\beta_i\}_{i=L+1}^{L+\ell_j})$;
    $L = L + \ell_j$;
    $C \leftarrow C \cup S$;
Output C

Multi-stage framework

APPROXGREED yields effective reduction on the number of function evaluations in LAZYGREED, however, the complexity of each function evaluation could still be so high that the greedy procedure is rendered impractical. To address this issue, we propose an approach, MULTGREED, that utilizes classes of simple surrogate functions which could be applied to a broad range of submodular functions. The idea is to optimize a series of surrogate (proxy) functions instead of optimizing the target function f.

Given a sequence $\{\beta_i\}_{i=1}^{\ell}$, a set of cardinality constraints $\{\ell_1, \dots, \ell_J\}$ such that $\sum_{j=1}^{J} \ell_j = \ell$ and $\ell_j > 0, \forall j$, and a corresponding set of J surrogate (proxy) submodular functions $\{f_j\}_{j=1}^{J}$, we define our framework MULTGREED as shown in Algorithm 9. The series of the surrogate functions should be designed in increasing order of complexity, and at the last stage of MULTGREED, f_J can even be the target function f. The algorithm should typically start with a computationally simple surrogate submodular function f_1 (which could even be modular). Since the surrogate functions f_j's are designed to be computationally cheaper than the target function f, and since APPROXGREED is applied instead of LAZYGREED in each stage, we are guaranteed to achieve an overall reduction in computation. In practice

(see Section 6.4 and 6.5), we often observe an instance of MultGreed with $J = 2$ suffices to yield good enough performance and complexity reduction as well, though our results are much more general.

Pruning: In addition to the above two schemes, it is also desirable to prune out items of the ground set that will never be chosen anyway, especially for large-scale data set. This is commonly done for submodular *minimization* [Fujishige, 2005,Iyer et al., 2013b]. Arbitrary pruning procedures, however, can significantly weaken the theoretical guarantee for Problem 6.2. We introduce here a simple new method that can prune away items without a corresponding performance loss. Consider the sequence of items $\{u_1, \ldots, u_n\}$ ordered non-increasingly in terms of their gain conditioned on all other items, i.e., $f(u_1|V \setminus u_1) \geq \cdots \geq f(u_n|V \setminus u_n)$. For an instance of Problem 6.2 with cardinality constraint ℓ, we have the following Lemma:

Lemma 5. LazyGreed *applied on the reduced ground set* $\hat{V} = \{j \in V | f(j) \geq f(u_\ell|V \setminus u_\ell)\}$ *is equivalent to that applied on the ground set* V*.*

The proofs for all the results in this chapter are deferred to Appendix. This procedure can easily be implemented in parallel, since $f(j)$ and $f(j|V \setminus j)$ can be computed independently for all $j \in V$. The pruning procedure is optional and is applicable only when the complexity of evaluating $f(u|V \setminus u)$ is no greater than that of $f(u)$. This is the case, for example, in our graph-based submodular functions. It is not true, however, for the entropy-based functions, nor the log-determinant style functions. Otherwise, the complexity of the pruning procedure could potentially even dominate that of LazyGreed, rendering it useless. MultGreed may optionally start with this ground set pruning step, but it does not influence our analysis.

Our analysis of Algorithm 9 is given in Section 6.2, while in Section 6.3, we illustrate examples on how to design surrogate functions. In Section 6.4, we shall instantiate our framework and provide recipes for choosing the parameters of MultGreed for several submodular functions which occur as models in real world applications.

6.2 Analysis

In this section, we formally analyze the methods presented in Section 6.1. We first define several crucial constructs that will facilitate this analysis.

Greedy ratio: We define a new construct we call the *greedy ratio* that will quantify the performance of a given instance of MULTGREED, which is characterized by the parameters: $\{f_j\}_{j=1}^{J}$, $\{\ell_j\}_{j=1}^{J}$, $\{\beta_i\}_{i=1}^{\ell}$. Guidelines on how to design the parameters of the multi-stage framework for several natural instances of useful submodular functions are given in Sections 6.3 and 6.4, but for now assume they are given. Let $s_1, \ldots, s_\ell$ be the sequence of items selected by the instance of MULTGREED. Let $S_i = \{s_1, \ldots, s_i\}$, be a set element of the chain $S_1 \subset S_2 \subset \cdots \subset S_\ell$, with $S_0 = \emptyset$.

Define the *individual greedy ratio* α_i for $i = 1, \ldots, \ell$ as:

$$\alpha_i = \frac{\max_{u \in V} f(u|S_{i-1})}{f(s_i|S_{i-1})} \tag{6.3}$$

Each α_i captures the ratio of the marginal gain of the greedily selected element to the marginal gain of the element s_i selected by MULTGREED. Therefore, α_i is a function of both the target function f but also, indirectly via ordered list $(s_1, s_2, \ldots, s_i)$, all of the remaining parameters $\{f_i\}_{i=1}^{K}$, $\{\ell_i\}_{i=1}^{K}$, $\{\beta_i\}_{i=1}^{\ell}$. Also, since $\max_{u \in V} f(u|S_{i-1}) \geq f(s_i|S_{i-1})$, we have that $\alpha_i \geq 1, \forall i$. Moreover, since under APPROXGREED we have $f(s_i|S_{i-1}) \geq \beta_i f(u|S_{i-1})$ for all $u \in V \setminus S_{i-1}$, it follows that $\alpha_i \leq 1/\beta_i$ for each i.

The list $\{\alpha_i\}_{i=1}^{\ell}$ collectively measures the quality of the multi-stage framework. We therefore define the *greedy ratio* α to be an aggregation of the individual greedy ratios. While there are many ways of aggregating, the harmonic mean, as we will show, provides the tightest characterization. We thus define the *greedy ratio* α as:

$$\alpha = \frac{\ell}{\sum_{i=1}^{\ell} 1/\alpha_i} \tag{6.4}$$

The greedy ratio, as we shall see, will provide a tight approximation guarantee. Ideally, we would like to have each individual greedy ratio $\alpha_i = 1$ for all i, and thus a greedy ratio of

$\alpha = 1$. In particular, our strategy for choosing surrogate functions and other parameters is to induce a greedy ratio that is as small as possible.

Curvature: Another important construct we shall need is the curvature (defined in Section 1.4.4). Given a submodular function f, recall that $\kappa_f(S)$ as the curvature of f with respect to a set S is defined as follows:

$$\kappa_f(S) = 1 - \min_{v \in V} \frac{f(v|S \setminus v)}{f(v)}. \tag{6.5}$$

$\kappa_f(S)$ lies in the range of $[0, 1]$, and is monotonically non-decreasing in S. It measures the distance of f from modularity and $\kappa_f = 0$ if and only if f is modular (or additive, i.e., $f(S) = \sum_{i \in S} f(i)$). The total curvature [Conforti and Cornuejols, 1984] κ_f is then $\kappa_f(V)$.

We now provide our main result:

Theorem 13. *Given a target submodular function f with total curvature κ_f, an instance of* MultGreed *with greedy ratio α is guaranteed to obtain a set S_ℓ s.t.*

$$\begin{aligned} \frac{f(S_\ell)}{f(S^{OPT})} &\geq \frac{1}{\kappa_f}\big(1 - (1 - \frac{1}{\alpha\ell})^{\ell\kappa_f}\big) \geq \frac{1}{\kappa_f}(1 - e^{-\frac{\kappa_f}{\alpha}}) \\ &\geq (1 - e^{-\frac{1}{\alpha}}) \end{aligned}$$

Conversely, for any value of $\alpha \geq 1$ and $\kappa_f \in [0, 1]$, there exists a submodular f with the total curvature κ_f, on which an instance of MultGreed *with the greedy ratio α achieves an approximation factor $\frac{1}{\kappa_f}\big(1 - (1 - \frac{1}{\alpha\ell})^{\ell\kappa_f}\big)$.*

Theorem 13 states that MultGreed's guarantee is quantified tightly by the greedy ratio α. Moreover, the bound is, indirectly via α, dependent on all the parameters $\{f_i\}_{i=1}^K$, $\{\ell_i\}_{i=1}^K$, $\{\beta_i\}_{i=1}^\ell$ of MultGreed. Theorem 13 generalizes bound $\frac{1}{\kappa_f}(1 - e^{-\kappa_f})$ [Conforti and Cornuejols, 1984] when $\alpha = 1$. By accounting for curvature, the bound $\frac{1}{\kappa_f}(1 - e^{-\kappa_f})$ itself generalizes the well-known result of $1 - 1/e$ for LazyGreed on Problem 6.2. Also, as an immediate corollary of Theorem 13, we obtain the theoretical guarantee for ApproxGreed in terms $\{\beta_i\}_{i=1}^\ell$.

Lemma 6. *Given a submodular function f with total curvature κ_f,* ApproxGreed$(f, \ell, \{\beta\}_{i=1}^{\ell})$ *is guaranteed to obtain a set S_ℓ: (here $\bar{\beta} = 1/\ell \sum_{i=1}^{\ell} \beta_i$)*

$$\begin{aligned}\frac{f(S_\ell)}{f(S^{OPT})} &\geq \frac{1}{\kappa_f}(1 - (1 - \frac{\bar{\beta}}{\ell})^{\ell\kappa_f}) \geq \frac{1}{\kappa_f}(1 - e^{-\kappa_f\bar{\beta}}) \\ &\geq (1 - e^{-\bar{\beta}}),\end{aligned}$$

If the $\{\beta_i\}_{i=1}^{\ell}$ are set as $\beta_i = c + \frac{1-c}{\ell} i$ with $0 \leq c \leq 1$ (c.f. Section 6.1), we have $\bar{\beta} \geq \frac{1+c}{2} \geq \frac{1}{2}$. Hence, this choice endows ApproxGreed with a solution having a factor no worse than $1 - e^{-1/2} \approx 0.39$.

The performance loss in MultGreed comes from two sources, namely the approximate greedy procedure and the surrogate functions. To simplify our analysis, we henceforth utilize only the exact greedy algorithm, $(\forall i, \beta_i = 1)$. It should be clear, however, that our results will immediately generalize to the approximate greedy case as well.

The greedy ratio α is the harmonic mean of the values $\{\alpha_i\}_{i=1}^{\ell}$ that themselves can be partitioned into J blocks based on the J stages of MultGreed. For $j = 1 \ldots J$, define $L_j = \sum_{j'=1}^{j} \ell_{j'}$, and let $I_j = \{L_{j-1} + 1, L_{j-1} + 2, \ldots, L_j\}$ be the set of ℓ_j indices for the j^{th} block. Each stage j provides a bound on the greedy ratio since $\alpha \leq \ell / \sum_{i \in I_j} 1/\alpha_i$. As a particularly simple example, if the target function f itself were to be utilized as the surrogate in the j^{th} stage for ℓ_j items, then each corresponding greedy ratio has value $\alpha_i = 1$ leading to the bound $\alpha \leq \ell/\ell_j$. Therefore, from the perspective of this upper bound, one is afforded the opportunity to design each stage semi-independently. On the other hand, to achieve a given desired α, it is not possible to design the stages entirely independently since the individual greedy ratios interact within the harmonic mean.

Generalization to knapsack constraint: Besides its flexibility in giving theoretical analysis of MultGreed, the greedy ratio can work in a broader scenario. Consider a more general formulation of Problem 6.2 as:

$$\max_{c(S) \leq b, S \subseteq V} f(S) \tag{6.6}$$

where $c(S) = \sum_{v \in S} c(v)$ with $c(v) \geq 0$ being the cost of $v \in V$ and b is the budget constraint. The problem becomes maximizing a monotone submodular function under a knapsack constraint (see Problem 4). Note many problems in machine learning applications, including sensor placement [Leskovec et al., 2007], document summarization [Lin and Bilmes, 2011], social networks [Singer, 2012] and speech data subset selection (see Chapter 2), are formulated in this form. Recall that a modified variant of the greedy algorithm (Alg. 5) solves this problem with a factor $\frac{1}{2}(1 - e^{-1})$ [Krause and Guestrin, 2005b]. Here, we call this variant the *knapsack greedy algorithm*

The knapsack greedy algorithm differs from LazyGreed in two aspects: (a) it, in each iteration, greedily selects the element that maximizes the marginal gain normalized by its cost, i.e., finding $s_i \in \text{argmax}_{u \in V \setminus S_{i-1}} \frac{f(u|S_{i-1})}{c(u)}$; (b) it compares the final solution of the greedy algorithm with the maximum singleton value with cost under the budget constraint B, i.e., $\max_{e \in V, c(e) \leq B} f(e)$, and outputs the maximum of the two. The multi-stage framework designed to solve Problem 6.2 can be adapted to incorporate the above two modifications and applied to Problem 6.6. Assume an instance of the multi-stage procedure stops at iteration N, we denote the chain of items selected as $\{s_1, \ldots, s_N\}$, with $S_i = \{s_1, \ldots, s_i\}$ for $i = 1, \ldots, N$. We generalize the definition of the individual greedy ratio as:

$$\alpha_i = \frac{\max_{u \in V \setminus S_{i-1}} \frac{f(u|S_{i-1})}{c(u)}}{\frac{f(s_i|S_{i-1})}{c(s_i)}} \tag{6.7}$$

for $i = 1, \ldots, N$. And define the *knapsack greedy ratio* as

$$\alpha = \frac{B'}{\sum_{j=1}^{N} \frac{c(s_j)}{\alpha_j}} \tag{6.8}$$

where $B' = \sum_{i=1}^{N} c(s_i)$. It is worth pointing out the previously defined greedy ratio is a special case of the knapsack greedy ratio, where $c(v) = 1$ for $v \in V$.

Theorem 14. *Given a target submodular function f, an instance of* MultGreed *with knapsack greedy ratio α is guaranteed to obtain a set S s.t.*

$$f(S) \geq \frac{1}{2}(1 - e^{-\frac{1}{\alpha}}) f(S^{OPT}) \tag{6.9}$$

where $S^{OPT} \in \text{argmax}_{S \subseteq V, c(S) \leq B} f(S)$.

Theorem 14 generalizes the approximation factor $\frac{1}{2}(1-e^{-1})$ [Krause and Guestrin, 2005b] where the knapsack greedy ratio is 1.

Generalization to submodular set cover problem: Closely related to Problem 6.6, the submodular set cover problem [Wolsey, 1982] is formulated as:

$$\min_{f(S) \geq f(V), S \subseteq V} c(S) \tag{6.10}$$

where f is a monotone submodular function and the same as before, $c(S) = \sum_{v \in S} c(v)$ with $c(v) \geq 0$ being the cost of item v. The same knapsack greedy algorithm solves Problem 6.10 with log factor approximation guarantee [Wolsey, 1982]. Therefore, the same multi-stage greedy framework that solves Problem 6.6 can be carried over to solve Problem 6.10. Given the observation that Problem 6.6 and 6.10 are duals of each other in that solution for one problem can be efficiently transformed to a solution for the other with bi-criterion approximation guarantee [Iyer and Bilmes, 2013], we obtain the following result.

Theorem 15. *An instance of* MultGreed *that solves Problem 6.6 with the knapsack greedy ratio* α *returns a solution that can be transformed to a solution* S *for Problem 6.10 such that*

$$c(S) \leq c(S^{OPT}) \text{ and } f(S) \geq \frac{1}{2}(1 - e^{-\frac{1}{\alpha}}) f(V), \tag{6.11}$$

where $S^{OPT} \in \text{argmax}_{S \subseteq V, f(S) \geq f(V)} c(S)$

The solution only satisfies approximate feasibility to Problem 6.10. Theorem 15 shows that an instance of the multi-stage knapsack greedy algorithm provides constant factor bi-criterion approximation guarantee, although Problem 6.10 does not admit any constant-factor approximation algorithms [Feige, 1998].

6.3 Surrogate Functions

In this section, we investigate the interplay between the greedy ratio and several choices of surrogate functions for classes of submodular functions which appear often in practice. Since

providing bounds on the performance of each stage individually implies an upper bound on the greedy ratio, we shall restrict ourselves to the analysis of the surrogate function at a given stage j, and the final performance guarantee is easily obtained by combining the guarantees for the different stages.

Uniform Submodular Mixtures: We first consider a class of submodular functions that can be represented as

$$f(S) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} f_t(S), \tag{6.12}$$

where $|\mathcal{T}| > 1$, and f_t is monotone submodular $\forall t \in \mathcal{T}$. We name this class *uniform submodular mixtures* as they are similar to the submodular mixtures previously defined in the context of learning [Lin and Bilmes, 2012]. They are also similar to the *decomposable* submodular functions of [Stobbe and Krause, 2010] but without the requirement that $f_t(S)$ be a non-decreasing concave function composed with a non-negative modular function. A number of natural submodular functions (e.g., facility location function, feature based submodular function, saturated coverage function, set cover function, etc.) belong to this class.

The complexity of evaluating f is determined both by $|\mathcal{T}|$ and the complexity of evaluating individual f_t's. Given such an f, a natural class of random surrogates takes the form

$$\hat{f}^{\text{sub}}(S) = \frac{1}{|\mathcal{T}'|} \sum_{i \in \mathcal{T}'} f_t(S), \tag{6.13}$$

where $\mathcal{T}' \subseteq \mathcal{T}$, and $\mathcal{T}'$ is generated by sampling individual elements from $\mathcal{T}$ with probability p. As p decreases, so does the complexity of evaluating $\hat{f}^{\text{sub}}$ but at the cost of a worse approximation to f. Applying a random function $\hat{f}^{\text{sub}}$ derived in this way to MultGreed, and assuming $|f_t(S)| \leq \mathcal{B}, \forall t \in \mathcal{T}, S \subseteq V$, we obtain:

Lemma 7. *Using the surrogate uniform mixture $\hat{f}^{sub}$ for stage j in* MultGreed *gives individual greedy ratios of*

$$1 \leq \alpha_i \leq \frac{1}{1-\epsilon}, \forall i \in I_j, \tag{6.14}$$

with probability $1-\delta$*, where* $\delta = (1 - 5n^{\ell} e^{-\frac{np(g^{\ell})^2\epsilon^2}{64B^2}})$ *and* $g^{\ell} = \max_{u \in V \setminus S_{\ell-1}} f(u|S_{\ell-1}) > 0$.

Fixing δ, a smaller value of probability p yields a higher value of ϵ, weakening the bound on each α_i. Fixing both δ and ϵ, increases in the ground set size $n = |V|$ could yield a choice of surrogate function $\hat{f}^{\text{sub}}$ having a smaller sampling probability p and thus that is easier to evaluate. More importantly, fixing δ and p, ϵ can be made arbitrarily close to 0 for n sufficiently large, a result that is of great interest for very large-scale problems. We shall use this result to provide bounds for several instances of submodular functions in Section 6.4

Modular Upper bounds: We next focus on a class of surrogate functions applicable to general submodular functions. Given a submodular function f, its simple modular upper bound is given as

$$\hat{f}^{\text{mod}}(S) = \sum_{s \in S} f(s). \tag{6.15}$$

For some submodular functions such as entropy (including Gaussian entropy and the log det functions used for DPPs) or mutual information, evaluating $\hat{f}^{\text{mod}}(S)$ is very easy, while evaluating $f(S)$ might sometimes even require computation exponential in $|S|$. Though extremely simple, this class nevertheless can act as an efficient class of surrogate functions especially useful when the target function is not very curved. $\hat{f}^{\text{mod}}$ is not only easy to optimize exactly, but it has previously been considered as a surrogate for various other forms of submodular optimization. The curvature κ_f, by definition, measures how close f is to being modular. If a modular surrogate function $\hat{f}^{\text{mod}}$, for general submodular function f, is applied within MultGreed, we can thus bound the individual greedy ratio via the curvature:

Lemma 8. *Using the modular upper bound as a surrogate function, it holds that*

$$1 \leq \alpha_i \leq \frac{1}{1 - \kappa_f(S_{i-1})}, \forall i \in I_j$$

Unsurprisingly, we see that the less curved the target function f is, the tighter bound on α_i's, and the better $\hat{f}^{\text{mod}}$ performs as a surrogate. In particular, if f is modular, i.e., $\kappa_f = 0$, then, all individual greedy ratio α_i's are tightly bounded as 1. Lemma 8 also implies that the bound of the individual greedy ratio weakens as i increases, since $\frac{1}{1-\kappa(S_{i-1})}$ increases with i. Therefore, this modular proxy, if applied, is best done in the first (or at most early) stages of MultGreed.

Graph based Submodular functions: We focus next on a class of submodular functions based on an underlying weighted graph and hence called *graph-based*. Many submodular functions used in machine learning applications belong to this class [Kolmogorov and Zabih, 2004, Wei et al., 2013, Liu et al., 2013, Lin and Bilmes, 2011]. These functions require $O(n^2)$ time to compute and store, which is not feasible for large n.

To form surrogates for the class of graph-based submodular functions, a natural choice is to utilize spanning subgraphs of the original graph. One choice is the k-nearest neighbor graph (k-NNG), defined as the spanning subgraph formed with each vertex $v \in V$ connected only to its k most similar neighbors (under the similarity score given by the edge weights). k-NNG has found great utilities in many machine learning applications [Shah et al., 2011, Boiman et al., 2008]. We write $\hat{f}^{\text{k-NNG}}$ as the surrogate function defined on a k-NNG for a graph-based submodular function f. The sparsity of the k-NNG depends on the value of k. The denser the graph (higher k), the costlier both the function evaluations and the memory complexity becomes. In Section 6.4, surprisingly we will show that $\hat{f}^{\text{k-NNG}}$, even for k as sparse as $O(\log n)$, can be good enough for certain graph-based functions.

6.4 Instantiations with Real World Submodular functions

Given the previously defined machinery to analyze MultGreed, we now focus on a broad range of submodular functions that appear as models in real world applications, and provide guidelines on how to design the surrogate functions as well as how to choose the size constraints. We investigate the following special cases: 1) the facility location function, 2)

saturated coverage function, 3) feature based function. We focus on analyzing the theoretical guarantees for these functions here and demonstrate the performance of some of these empirically in the next section.

Facility location function: Given a weighted graph $G = (V, E)$, with $w_{u,v}$ the edge weight (i.e., similarity score) between vertices u and v for $u, v \in V$, the (uncapacitated) facility location function is defined as

$$f_{\text{fac}}(S) = \sum_{v \in V} \max_{u \in S} w_{u,v}. \tag{6.16}$$

Define $\hat{w}$ as k-NNG counterpart of w, i.e., $\hat{w}_{i,j} = w_{i,j}$ if j is among the k nearest neighbor of i, and $\hat{w}_{i,j} = 0$, otherwise. To establish that $\hat{f}_{\text{fac}}^{\text{k-NNG}}$, even with very sparse k, is a good approximation of f_{fac}, we rely on a key observation: $\max_{j \in S} w_{i,j} = \max_{j \in S} \hat{w}_{i,j}$ holds if the set S contains at least one item that is among the k nearest neighbor of i. Thus, showing that $\hat{f}_{\text{fac}}^{\text{k-NNG}}(S) = f_{\text{fac}}(S)$ is equivalent as showing that the set S contains at least one item that is among the k nearest neighbor of item i for any $i \in V$. Let's denote $\vec{w}_i = \{w_{i,1}, \ldots, w_{i,n}\}$ as the vector containing the weights on all edges out of the vertex i. To this end, we assume that the ranking of any item $j \in V$ among the vector $\vec{w}_i$ for any $i \in V$ is uniformly distributed over $\{1, 2, \ldots, n\}$ and that the ranking of j in one weight vector $\vec{w}_i$ is independent of its ranking in another.

Lemma 9. *For the facility location function, we have:*

$$\hat{f}_{fac}^{k\text{-}NNG}(S) = f_{fac}(S), \forall S \subseteq V \ s.t. \ |S| \geq m, \tag{6.17}$$

with probability at least $(1 - \theta)$*, and the sparsity of the k-NNG being at least*

$$k = n[1 - (\frac{\theta}{n})^{\frac{1}{m}}]. \tag{6.18}$$

Assuming that m, n are in the same order and that θ is a constant, we have $\lim_{n \to \infty} n[1 - (\frac{\theta}{n})^{\frac{1}{m}}] = O(\log n)$. The Lemma implies that with high probability, $\hat{f}_{\text{fac}}^{\text{k-NNG}}$ and f_{fac} share

the same function value for any sets of size greater than some threshold m, where the k-NNG can be as sparse as $k = O(\log n)$.

By Lemma 9, $\hat{f}^{\text{k-NNG}}_{\text{fac}}$ alone provides a good approximation for f_{fac}. It thus suffices, in this case, to apply a single-stage greedy algorithm (MultGreed with $J = 1$) using $\hat{f}^{\text{k-NNG}}_{\text{fac}}$ as the surrogate. As a concrete example, consider an instance of the procedure with $\theta = 0.05$, $n = 10^6$, $k = 0.009n$, and $\ell = 0.1n$. Then, Lemma 9 implies that with probability 95%, $\hat{f}^{\text{k-NNG}}_{\text{fac}}(S) = f_{\text{fac}}(S)$ holds for any $|S| \geq 0.00186n$, giving an individual greedy ratio of $\alpha_i = 1$ for $0.00186n \leq i \leq 0.1n$. The greedy ratio α, defined as the harmonic mean of $\{\alpha_i\}_{i=1}^{\ell}$, is then bounded as $\alpha \leq 1.02$, which guarantees a solution in this instance close to optimum, thanks to Theorem 13.

Saturated coverage function: Successfully applied in document summarization [Lin and Bilmes, 2011], the saturated coverage function is another subclass of graph-based submodular functions, defined as

$$f_{\text{sat}}(S) = \sum_{v \in V} \min\{\sum_{u \in S} w_{v,u}, \xi \sum_{u \in V} w_{v,u}\}, \tag{6.19}$$

where $0 < \xi \leq 1$ is a hyperparameter that determines the saturation ratio. The class of uniform submodular mixtures includes f_{sat}. In this case, we can construct a two-stage greedy algorithm, where the modular upper bound $\hat{f}^{\text{mod}}$ and a sampling based function $\hat{f}^{\text{sub}}$ (with sampling probability p) are used as the two surrogates.

Lemma 10. *Given the saturated coverage function, an instance of* MultGreed *with the size constraints* $\ell_1 = \lfloor \frac{n\xi}{(1-\xi)\gamma+\xi} \rfloor$ *and* $\ell_2 = \max\{0, \ell - \ell_1\}$ *(where* $\gamma = \frac{\max_{u,v} w_{u,v}}{\min_{(u,v)\in E(G)} w_{u,v}}$*, assuming all extent graph edges are positively weighted) yields a solution with the individual greedy ratios*

$$\alpha_i = 1, \text{ for } i = 1, \ldots, \ell_1$$

And with probability $1 - \delta$*,*

$$1 \leq \alpha_i \leq \frac{1}{1-\epsilon}, \text{ for } i = \ell_1 + 1 \ldots, \ell$$

where $\delta = (1 - 5n^{\ell}e^{-\frac{np(g^{\ell})^2\epsilon^2}{64B^2}})$ *and* $g^{\ell} = \max_{u \in V \setminus S_{\ell-1}} f(u|S_{\ell-1}) > 0$.

A main intuition of this result, is that f_{sat} is modular up to set of size ℓ_1. Hence it makes sense to use $\hat{f}^{\text{mod}}$ for these cases. Similarly for the second stage, it is reasonable to use $\hat{f}^{\text{sub}}$ with an appropriate p.

Feature based submodular function: Successfully applied in the case studies on speech data subset selection (Chapter 2) and batch active learning (Chapter 4), the feature based submodular function can also be efficiently optimized using MultGreed. Recall that the feature based submodular function has the following form:

$$f_{\text{fea}}(S) = \sum_{u \in U} w_u g(c_u(S)), \tag{6.20}$$

where g is concave, U is a set of "features", $w_u > 0$ is the weight for the feature u, and $c_u(S) = \sum_{v \in S} c_u(s)$ is a non-negative modular score for feature $u \in U$ in set S, with $c_u(v)$ measuring the degree to which item u possesses feature u. Here, we consider a special case of the feature based function with the concave function of the form $g(x) = x^a$ for $0 < a \leq 1$ and the weights $w_u = 1, \forall u \in U$. Again, f_{fea} is a member of the class of uniform submodular mixtures. The curvature of f_{fea} is governed by the curvature of the concave function g and thus is determined by a. We can construct a two-stage procedure similar to that for f_{sat}, where we optimize over $\hat{f}^{\text{mod}}$ and $\hat{f}^{\text{sub}}$ with a suitable choice of the sampling probability p.

Lemma 11. *Given the feature based submodular function, an instance of* MultGreed *with the size constraints being* ℓ_1 *and* ℓ_2*, yields a solution with the individual greedy ratio bounded as:*

$$1 \leq \alpha_i \leq O(i^{1-a}), \text{ for } i = 1, \ldots, \ell_1$$

And with probability $1 - \delta$*,*

$$1 \leq \alpha_i \leq \frac{1}{1-\epsilon}, \text{ for } i = \ell_1 + 1 \ldots, \ell$$

where $\delta = (1 - 5n^{\ell}e^{-\frac{np(g^{\ell})^2\epsilon^2}{64B^2}})$ *and* $g^{\ell} = \max_{u \in V \setminus S_{\ell-1}} f(u|S_{\ell-1}) > 0$.

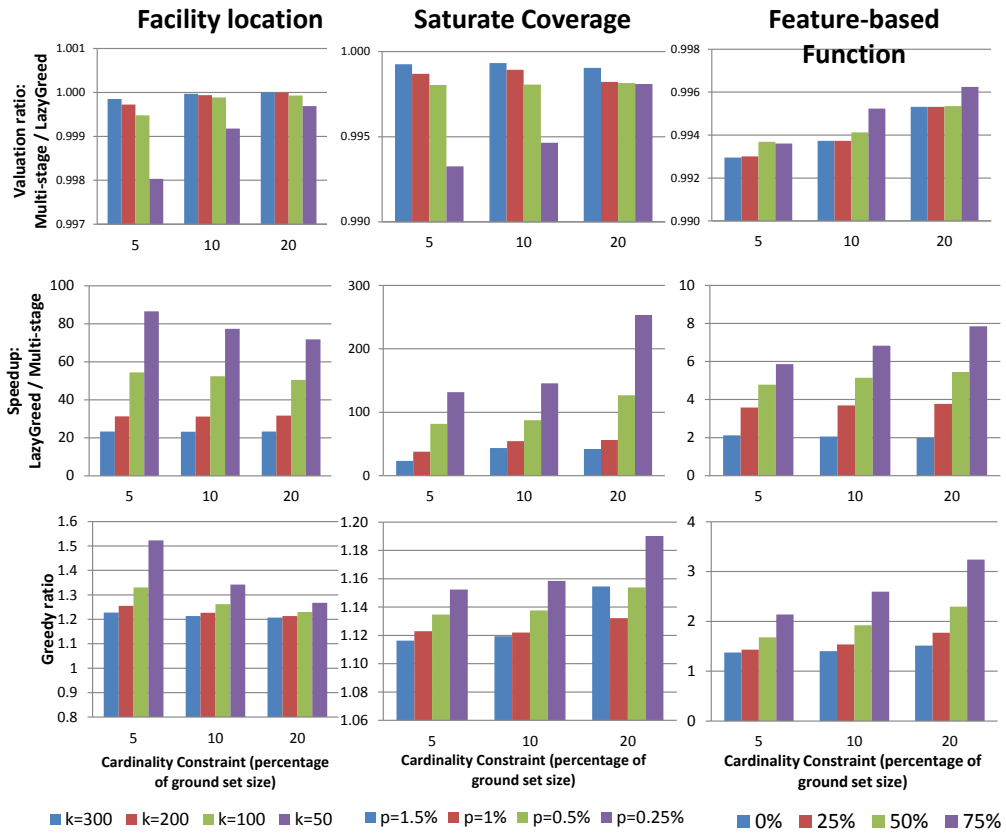


Figure 6.1: A comparison of the function values (top row), the running time (middle row), the greedy ratio (bottom row) between lazy greedy and our multi-stage approach, under different choices of the surrogate function for f_{fac} (left column), f_{sat} (middle column), and f_{fea} (right column).

The lemma implies that with an appropriate choice of the sampling probability p for $\hat{f}^{\text{sub}}$, the performance loss in the second stage could be negligible. However, there is some performance loss introduced by the first stage, depending on a and ℓ_1. The choices for ℓ_1 and ℓ_2 determine the tradeoff between loss of the performance guarantee and the computational reduction: larger ℓ_1 is chosen when computation is critical or when g is less curved (larger values of a), while larger ℓ_2 is chosen when algorithmic performance is the priority or g is more curved (smaller values of a).

6.5 Experiments

We empirically test the performance of MultGreed for three of the submodular functions considered above. We address the following questions: 1) how well does MultGreed perform compared to LazyGreed, 2) how much relative time reduction can be achieved, 3) how well does the greedy ratio perform as a quality measure, 4) how well does the framework scale to massive data sets. We run experiments on two scenarios: 1) simulations with medium sized data, 2) real world speech data selection on millions of ground elements.

6.5.1 Simulations

All simulations are performed on the same data with size $|V| = 20,000$, formed by randomly sampling from a large speech recognition training corpus (the "Fisher" corpus). Each sample pair has a similarity score, and the graph-based submodular functions f_{fac} and f_{sat} are instantiated using the corresponding similarity matrix. A set of features $\mathcal{F}$ sized $|\mathcal{F}| \approx 75000$ is derived from the same data to instantiate f_{fea}. In all runs of MultGreed, we set $\{\beta_i\}_{i=1}^{\ell}$ using the schedule $\beta_i = c + \frac{(i-1)(1-c)}{\ell}$ with $c = 0.5$. Performance of MultGreed and LazyGreed is measured by the function valuations and the wall-clock running time. The optional stage of pruning is performed only for f_{fea}, but not for f_{fac} f_{sat}, since both f_{fac} and f_{sat} are very curved functions, for which the pruning procedure cannot effectively remove items.

For f_{fac}, use one stage with surrogates $\hat{f}^{\text{k-NNG}}$ with $k \in \{50, 100, 200, 300\}$. MultGreed gives about a 20-80 times speedup over LazyGreed with at least 99.8% of the standard greedy solution (first column of Fig. 6.1). For f_{sat}, the saturation ratio ξ is set as 0.25. Two stages using surrogate functions $\hat{f}^{\text{mod}}$ and $\hat{f}^{\text{sub}}$ are applied, under size constraints $\ell_1 = \lfloor \frac{n\xi}{5(1-\xi)+\xi} \rfloor = 0.05n$, and $\ell_2 = \ell - \ell_1$. We test $\hat{f}^{\text{sub}}$ with various sampling probabilities: $p \in \{0.25\%, 0.5\%, 1\%, 1.5\%\}$. The results (2nd column of Fig. 6.1) show a speedup of up to 250 with at least 99.25% the quality of LazyGreed. Next, for f_{fea}, we set g to be the square root function. Two stages of surrogates $\hat{f}^{\text{mod}}$ and $\hat{f}^{\text{sub}}$ are applied. $\hat{f}^{\text{sub}}$ is defined on a randomly selected feature subset of size 37,500. We test with different com-

	5%	10%	20%	all
Averaged Random	38.2	35.1	34.4	
Histogram-Entropy	37.6	34.2	fail	31.0
Multi-stage Submodular	**37.3**	**34.1**	**32.7**	

Table 6.1: Word error rates under averaged random, histogram-entropy, and the multi-stage submodular chosen subsets at various sized percentages (lower the better). Histogram-entropy result for the 20% condition is not available due to its objective's saturation after 10%.

binations of size constraints ℓ_1 and ℓ_2 by setting $\ell_1 \in \{0, 0.25\ell, 0.5\ell, 0.75\ell\}$ with $\ell_2 = \ell - \ell_1$. This gives about a 2-8 times speedup with at least 99.3% of LazyGreed quality (right column of Fig 6.1). Empirically, the greedy ratio is very tight since it is always close to 1. For most cases, it is a good indicator of the performance for function valuations, since lower values of α always lead to higher performance of MultGreed. For f_{fac} and f_{sat}, the speedup reported does not include the potentially significant additional complexity reduction on graph-construction. Especially for f_{fac}, efficient algorithms exist for fast (approximate) construction of the k-NNG [Beygelzimer et al., 2006].

6.5.2 Speech Data Subset Selection

We next test the performance of MultGreed on a very large-scale problem, where running even LazyGreed is infeasible. We address the problem of speech data subset selection (Chapter 2): given a massive (speech) data set for training automatic speech recognition (ASR) systems, we wish to select a representative subset that fits a given budget (measured in total time) and train a system only on the subset. Problem 6.6 addresses this where the objective is the facility location function f_{fac}, and the pair-wise similarity between speech samples is computed by TF-IDF kernel as defined in Chapter 2. We subselect 1300 hours of conversational English telephone data from the "Switchboard", "Switchboard Cellular",

and "Fisher" corpora, which, in total, comprises 1,322,108 segments of speech (i.e., $|V| = n = 1,322,018$). The estimated running time of LazyGreed with f_{fac} on such large data is at least a week. Rendering the full $O(n^2)$ similarity matrix is even more impractical due to memory requirements. We here test MultGreed using $\hat{f}_{\text{fac}}^{\text{k-NNG}}$ with the sparsity of k-NNG set as $k = 1,000$. MultGreed, then, runs in only a few minutes, yielding a speedup of more than a thousand over LazyGreed! We measure the performance of the selection by the word error rate (WER) of the ASR system trained on the corresponding selected subset of the data. We test on different budget constraints (5%, 10% and 20% of the whole speech data). We compare our selection against two baseline selection methods: (1) averaged random method, where we randomly sample the data set at appropriate sizes, train different ASR systems for each set, and average their WER; (2) a non-submodular "histogram-entropy" based method, described in [Wu et al., 2007]. Table 6.1 illustrates that our framework yields consistently superior results to these baselines.

6.6 Discussion

Certain other domains may be applicable to the analysis introduced in this chapter. In the case of feature selection, for example, one may wish to optimize the mutual information function $f_{\text{mi}} = I(X_S; C)$ which either is not submodular, or can become submodular by assuming that the random variables X_V are independent given C [Krause and Guestrin, 2005a]. In either case, however, the complexity of evaluating f_{mi} can be daunting, leading to previous work suggesting a tractable surrogate $\hat{f}_{\text{mi}}(S) = \sum_{v \in S} I(X_v; C) - \lambda \sum_{v,u \in S} I(X_v; X_u)$, where λ is a hyperparameter [Peng et al., 2005]. Under certain assumptions, this surrogate is in fact equivalent to the original [Balagani and Phoha, 2010]. Unnoticed by these authors, however, this function is submodular and non-monotone. We plan in the future to extend our framework to additionally handle such functions.

We would also like to extend these ideas to other submodular optimization scenarios, like submodular minimization [Fujishige and Isotani, 2011, Iyer et al., 2013a, Iyer et al., 2013b], and the class of optimization problems involving submodular constraints [Iyer and Bilmes,

2013] (which includes the submodular set cover, the submodular cost submodular set cover, and the submodular cost submodular knapsack). While many of the algorithms for these problems use proxy functions as surrogates, they often choose generic functions as proxies to obtain theoretical guarantees. It will be interesting to see if an intelligent design of surrogate functions, could yield better theoretical guarantees for real world problems.

Part II
DATA PARTITIONING

So far, we have mainly focused on data summarization paradigm for addressing the big data challenge. An alternative solution to this challenge is via parallelism or distributed computation as will be done in Part II. The goal of data partitioning is to split the data into multiple compute nodes such that the utility or the goodness of the data residing on each node is high. In some other cases, an intelligent data partitioning scheme aims to distribute data so as to minimize the communication and/or memory cost associated with each compute node. As we will see in Part II, these problems can be naturally formulated in terms of our novel submodular data partitioning problem. Similar to the data summarization paradigm, the success of submodular data partitioning paradigm significantly hinges on the appropriate utility model as well as efficient optimization algorithms. Given the success of the case studies in Part I, we are now confident that submodular functions are appropriate utility models for a number of machine learning tasks. Hence, our focus here is more on the optimization component of the problem. In Chapter 7, we study various forms submodular partitioning problems, for which we give efficient and/or tight approximation algorithms. In Chapter 8, we give a concrete case study that applies submodular partitioning framework to intelligently distributing data for parallel learning statistical models.

Chapter 7

MIXED ROBUST/AVERAGE SUBMODULAR PARTITIONING

This chapter studies the optimization algorithms for the two new mixed robust/average-case partitioning problems as previously defined in Section 1.5.6. For completeness of this Chapter, we redefine the problems as follows:

$$\max_{\pi\in\Pi}\Big[\bar{\lambda}\min_i f_i(A_i^\pi)+\frac{\lambda}{m}\sum_{j=1}^{m} f_j(A_j^\pi)\Big], \tag{7.1}$$

$$\min_{\pi\in\Pi}\Big[\bar{\lambda}\max_i f_i(A_i^\pi)+\frac{\lambda}{m}\sum_{j=1}^{m} f_j(A_j^\pi)\Big], \tag{7.2}$$

where $0 \leq \lambda \leq 1$, $\bar{\lambda} \triangleq 1-\lambda$, the set of sets $\pi = (A_1^\pi, A_2^\pi, \cdots, A_m^\pi)$ is an ordered partition of a finite set V (i.e, $\cup_i A_i^\pi = V$ and $\forall i \neq j, A_i^\pi \cap A_j^\pi = \emptyset$), and Π refers to the set of all ordered partitions of V into m blocks. In contrast to the notion of the partition often used in the computer science and mathematical communities, we clarify that an ordered partition π is fully characterized by both its constituent blocks $\{A_i^\pi\}_{i=1}^m$ as well as the ordering of the blocks. The parameter λ controls the objective: $\lambda = 1$ is the average case, $\lambda = 0$ is the robust case, and $0 < \lambda < 1$ is a mixed case. In general, Problems 7.1 and 7.2 are hopelessly intractable, even to approximate, but we assume that the $f_1, f_2, \cdots, f_m$ are all monotone non-decreasing (i.e., $f_i(S) \leq f_i(T)$ whenever $S \subseteq T$), normalized ($f_i(\emptyset) = 0$), and submodular (i.e., $\forall S, T \subseteq V$, $f_i(S) + f_i(T) \geq f_i(S \cup T) + f_i(S \cap T)$). These assumptions allow us to develop fast, simple, and scalable algorithms that have approximation guarantees, as is done in this chapter. When minimizing, submodularity naturally model notions of interacting costs and complexity, while when maximizing it readily models notions of diversity, summarization quality, and information. Hence, Problem 7.1 asks for a partition whose blocks each

Problem 7.1 (Max-(Min+Avg))	
	Approximation factor
$\lambda = 0$, BinSrch [Khot and Ponnuswami, 2007]	$1/(2m-1)$
$\lambda = 0$, Matching [Golovin, 2005]	$1/(n-m+1)$
$\lambda = 0$, Ellipsoid [Goemans et al., 2009]	$O(\sqrt{n}m^{1/4}\log n \log^{3/2} m)$
$\lambda = 1$, GreedWelfare [Fisher et al., 1978]	$1/2$
$\lambda = 0$, GreedSat*	$(1/2-\delta, \frac{\delta}{1/2+\delta})$
$\lambda = 0$, MMax*	$O(\min_i \frac{1+(\lvert A_i^{\hat{\pi}}\rvert-1)(1-\kappa_{f_i}(A_i^{\hat{\pi}}))}{\lvert A_i^{\hat{\pi}}\rvert\sqrt{m}\log^3 m})$
$\lambda = 0$, GreedMax†*	$1/m$
$0 < \lambda < 1$, GeneralGreedSat*	$\lambda/2$
$0 < \lambda < 1$, CombSfaSwp*	$\max\{\frac{\beta\alpha}{\lambda\beta+\alpha}, \lambda\beta\}$
$0 < \lambda < 1$, CombSfaSwp†*	$\max\{\min\{\alpha, \frac{1}{m}\}, \frac{\beta\alpha}{\lambda\beta+\alpha}, \lambda\beta\}$
$\lambda = 0$, Hardness	$1/2$ [Golovin, 2005]
$\lambda = 1$, Hardness	$1-1/e$ [Vondrák, 2008]

Table 7.1: Summary of our contributions and existing work on Problem 7.1.[2] See text for details.

(and that collectively) are a good, say, summary of the whole. Problem 7.2 on the other hand, asks for a partition whose blocks each (and that collectively) are internally homogeneous (as is typical in clustering). Taken together, we call Problems 7.1 and 7.2 *Submodular Partitioning*. We further categorize these problems depending on if the f_i's are identical to each other (*homogeneous*) or not (*heterogeneous*).[1] The heterogeneous case clearly generalizes the homogeneous setting, but as we will see, the additional homogeneous structure can be exploited to provide more efficient and/or tighter algorithms.

[1]Similar sub-categorizations have been called the "uniform" vs. the "non-uniform" case in the past [Svitkina and Fleischer, 2008, Goemans et al., 2009].

Special cases of Problem 7.1 have appeared previously in the literature. Problem 7.1 with $\lambda = 0$ is called *submodular fair allocation* (SFA), which has been studied mostly in the heterogeneous setting. When f_i's are all modular, the tightest algorithm, so far, is to iteratively round an LP solution achieving $O(1/(\sqrt{m}\log^3 m))$ approximation [Asadpour and Saberi, 2010], whereas the problem is NP-hard to $1/2+\epsilon$ approximate for any $\epsilon > 0$ [Golovin, 2005]. When f_i's are submodular, [Golovin, 2005] gives a matching-based algorithm with a factor $1/(n-m+1)$ approximation that performs poorly when $m \ll n$. [Khot and Ponnuswami, 2007] propose a binary search algorithm yielding an improved factor of $1/(2m-1)$. Another approach approximates each submodular function by its ellipsoid approximation (non-scalable) and reduces SFA to its modular version leading to an approximation factor of $O(\sqrt{n}m^{1/4}\log n \log^{3/2} m)$. These approaches are theoretically interesting, but they either do not fully exploit the problem structure or cannot scale to large problems. On the other hand, Problem 7.1 for $\lambda = 1$ is called *submodular welfare*. This problem has been extensively studied in the literature and can be equivalently formulated as submodular maximization under a partition matroid constraint [Vondrák, 2008]. It admits a scalable greedy algorithm that achieves a $1/2$ approximation [Fisher et al., 1978]. More recently a multi-linear extension based algorithm nicely solves the submodular welfare problem with a factor of $(1-1/e)$ matching the hardness of this problem [Vondrák, 2008]. It is worth noting that the homogeneous instance of the submodular welfare with $m = 2$ generalizes the well-known max-cut problem [Goemans and Williamson, 1995], where the submodular objective is defined as the submodular neighborhood function [Iyer and Bilmes, 2013]. As far as we know, Problem 7.1 for general $0 < \lambda < 1$ has not been studied in the literature.

When $\lambda = 0$, Problem 7.2 is studied as *submodular load balancing* (SLB). When f_i's are all modular, SLB is called *minimum makespan scheduling*. In the homogeneous setting, [Hochbaum and Shmoys, 1988] give a PTAS scheme ($(1+\epsilon)$-approximation algorithm which

[2]Results obtained in this paper are marked as $*$. Methods for only the homogeneous setting are marked as †.

[3]Results obtained in this paper are marked as $*$. Methods for only the homogeneous setting are marked as †.

Problem 7.2 (Min-(Max+Avg))	
	Approximation factor
$\lambda = 0$, BALANCED† [Svitkina and Fleischer, 2008]	$\min\{m, n/m\}$
$\lambda = 0$, SAMPLING [Svitkina and Fleischer, 2008]	$O(\sqrt{n \log n})$
$\lambda = 0$, ELLIPSOID [Goemans et al., 2009]	$O(\sqrt{n} \log n)$
$\lambda = 1$, GREEDSPLIT† [Zhao et al., 2004, Narasimhan et al., 2005]	2
$\lambda = 1$, RELAX [Chekuri and Ene, 2011a]	$O(\log n)$
$\lambda = 0$, MMIN*	$\max_i \frac{2\lvert A_i^{\pi^*}\rvert}{1+(\lvert A_i^{\pi^*}\rvert-1)(1-\kappa_{f_i}(A_i^{\pi^*}))}$
$\lambda = 0$, LOVÁSZ ROUND*	m
$0 < \lambda < 1$ GENERALLOVÁSZ ROUND*	m
$0 < \lambda < 1$, COMBSLBSMP*	$\min\{\frac{m\alpha}{m\bar{\lambda}+\lambda}, \beta(m\bar{\lambda}+\lambda)\}$
$0 < \lambda < 1$, COMBSLBSMP$^{\dagger *}$	$\min\{m, \frac{m\alpha}{m\bar{\lambda}+\lambda}, \beta(m\bar{\lambda}+\lambda)\}$
$\lambda = 0$, Hardness*	m
$\lambda = 1$, Hardness	$2 - 2/m$ [Ene et al., 2013]

Table 7.2: Summary of our contributions and existing work on Problem 7.2[3].

runs in polynomial time for any fixed ϵ), while an LP relaxation algorithm provides a 2-approximation for the heterogeneous setting [Lenstra et al., 1990]. When the objectives are submodular, the problem becomes much harder. Even in the homogeneous setting, [Svitkina and Fleischer, 2008] show that the problem is information theoretically hard to approximate within $o(\sqrt{n/\log n})$. They provide a balanced partitioning algorithm yielding a factor of $\min\{m, n/m\}$ under the homogeneous setting. They also give a sampling-based algorithm achieving $O(\sqrt{n/\log n})$ for the homogeneous setting. However, the sampling-based algorithm is not practical and scalable since it involves solving, in the worst-case, $O(n^3 \log n)$ instances of submodular function minimization each of which in general currently requires $O(n^5\gamma + n^6)$ computation [Orlin, 2009], where γ is the cost of a function valuation. Similar to Submodular Fair Allocation, [Goemans et al., 2009] applies the same ellipsoid approxi-

mation techniques leading to a factor of $O(\sqrt{n}\log n)$ [Goemans et al., 2009]. When $\lambda = 1$, Problem 7.2 becomes the *submodular multiway partition* (SMP) for which one can obtain a relaxation based 2-approximation [Chekuri and Ene, 2011a] in the homogeneous case. In the non-homogeneous case, the guarantee is $O(\log n)$ [Chekuri and Ene, 2011b]. Similarly, [Zhao et al., 2004, Narasimhan et al., 2005] propose a greedy splitting 2-approximation algorithm for the homogeneous setting. To the best of our knowledge, there does not exist any work on Problem 7.2 with general $0 < \lambda < 1$.

In contrast to Problems 7.1 and 7.2 in the average case (i.e., $\lambda = 1$), existing algorithms for the worst case ($\lambda = 0$) are not scalable. In this chapter, we close this gap, by proposing three new classes of algorithmic frameworks to solve SFA and SLB: (1) greedy algorithms; (2) semigradient-based algorithms; and (3) a Lovász extension based relaxation algorithm.

For SFA, when $m = 2$, we formulate the problem as non-monotone submodular maximization, which can be approximated up to a factor of $1/2$ with $O(n)$ function evaluations [Buchbinder et al., 2012]. For general m, we give a simple and scalable greedy algorithm (GreedMax), and show a factor of $1/m$ in the homogeneous setting, improving the state-of-the-art factor of $1/(2m-1)$ under the heterogeneous setting [Khot and Ponnuswami, 2007]. For the heterogeneous setting, we propose a "saturate" greedy algorithm (GreedSat) that iteratively solves instances of submodular welfare problems. We show GreedSat has a bi-criterion guarantee of $(1/2 - \delta, \delta/(1/2 + \delta))$, which ensures at least $\lceil m(1/2 - \delta)\rceil$ blocks receive utility at least $\delta/(1/2 + \delta)OPT$ for any $0 < \delta < 1/2$. For SLB, we first generalize the hardness result in [Svitkina and Fleischer, 2008] and show that it is hard to approximate better than m for any $m = o(\sqrt{n/\log n})$ even in the homogeneous setting. We then give a Lovász extension based relaxation algorithm (Lovász Round) yielding a tight factor of m for the heterogeneous setting. As far as we know, this is the first algorithm achieving a factor of m for SLB in this setting. For both SFA and SLB, we also obtain more efficient algorithms with bounded approximation factors, which we call majorization-minimization (MMin) and minorization-maximization (MMax).

Next we show algorithms to handle Problems 7.1 and 7.2 with general $0 < \lambda < 1$. We

first give two simple and generic schemes (CombSfaSwp and CombSlbSmp), both of which efficiently combines an algorithm for the worst-case problem (special case with $\lambda = 0$), and an algorithm for the average case (special case with $\lambda = 1$) to provide a guarantee interpolating between the two bounds. Given the efficient algorithms proposed in this paper for the robust (worst case) problems (with guarantee α), and an existing algorithm for the average case (say, with a guarantee β), we can obtain a combined guarantee in terms of α, β and λ. We then generalize the proposed algorithms for SLB and SFA to give more practical algorithmic frameworks to solve Problems 7.1 and 7.2 for general λ. In particular we generalize GreedSat leading to GeneralGreedSat, whose guarantee smoothly interpolates in terms of λ between the bi-criterion factor by GreedSat in the case of $\lambda = 0$ and the constant factor of $1/2$ by the greedy algorithm in the case of $\lambda = 1$. For Problem 7.2 we generalize Lovász Round to obtain a relaxation algorithm (GeneralLovász Round) that achieves an m-approximation for general λ. Motivated by the computational limitation of GeneralLovász Round we also give a simple and efficient greedy heuristic called GeneralGreedMin that works for the homogeneous setting of Problem 7.2.

We also demonstrate a number of applications of submodular partitioning in real-world machine learning problems. We show Problem 7.1 is applicable in distributed training of statistical models. Problem 7.2, on the other hand, is useful for data clustering, image segmentation, and computational load balancing. In this chapter we empirically show the efficacy of Problem 7.2 on an unsupervised image segmentation task. We defer the concrete case study of applying Problem 7.1 to distributed training statistical models to Chapter 8. As an outline of this chapter, we provide algorithms for SFA and SLB in Section 7.1. Algorithms for Problems 7.1 and 7.2 with general λ are given in Section 7.2. We demonstrate the applications of Problems 7.1 and 7.2 in Section 7.3. Experimental results are given in Section 7.4.

7.1 Robust Submodular Partitioning (Problems 7.1 and 7.2 when $\lambda = 0$)

Here, we give some useful notations to facilitate the description of this chapter: we define $f(j|S) \triangleq f(S \cup j) - f(S)$ as the gain of $j \in V$ in the context of $S \subseteq V$. Then, f is

submodular if and only if $f(j|S) \geq f(j|T)$ for all $S \subseteq T$ and $j \notin T$. Also, f is monotone iff $f(j|S) \geq 0, \forall j \notin S, S \subseteq V$. We assume w.l.o.g. that the ground set is $V = \{1, 2, \cdots, n\}$.

7.1.1 Approximation Algorithms for SFA (Problem 7.1 with $\lambda = 0$)

We first investigate a special case of SFA with $m = 2$. When $m = 2$, the problem becomes

$$\max_{A \subseteq V} g(A), \tag{7.3}$$

where $g(A) = \min\{f_1(A), f_2(V \setminus A)\}$ and is submodular thanks to Theorem 16.

Theorem 16. *If f_1 and f_2 are monotone submodular, $\min\{f_1(A), f_2(V \setminus A)\}$ is also submodular.*

All proofs for the theoretical results in this chapter are given in Appendix. Interestingly SFA for $m = 2$ can be equivalently formulated as unconstrained submodular maximization (see Problem 1 for more details). This problem has been well studied in the literature [Buchbinder et al., 2012, Dobzinski and Mor, 2015, Feige et al., 2011]. A simple bi-directional randomized greedy algorithm (see Algorithm 2) solves Eqn 7.3 with a tight factor of 1/2. Applying this randomized algorithm to solve SFA then achieves a guarantee of 1/2 matching the problem's hardness. However, the same idea does not apply to the general case of $m > 2$.

For general m, we approach SFA from the perspective of the greedy algorithms. Greedy is often the algorithm that practitioners use for combinatorial optimization problems since they are intuitive, simple to implement, and often lead to very good solutions. Here we introduce two variants of a greedy algorithm – GreedMax (Alg. 10) and GreedSat (Alg. 11), suited to the homogeneous and heterogeneous settings, respectively.

GreedMax: The key idea of GreedMax (see Alg. 10) is to greedily add an item with the maximum marginal gain to the block whose current solution is minimum. Initializing $\{A_i\}_{i=1}^m$ with the empty sets, the greedy flavor also comes from that it incrementally grows the solution by greedily improving the overall objective $\min_{i=1,\ldots,m} f_i(A_i)$ until $\{A_i\}_{i=1}^m$ forms a partition. Besides its simplicity, Theorem 17 offers the optimality guarantee.

Algorithm 10: GreedMax

1: Input: f, m, V.
2: Let $A_1 =, \ldots, = A_m = \emptyset$; $R = V$.
3: **while** $R \neq \emptyset$ **do**
4: $\quad j^* \in \text{argmin}_j f(A_j)$;
5: $\quad a^* \in \text{argmax}_{a \in R} f(a|A_{j^*})$
6: $\quad A_{j^*} \leftarrow A_{j^*} \cup \{a^*\}$; $R \leftarrow R \setminus a^*$
7: **end while**
8: Output $\{A_i\}_{i=1}^m$.

Theorem 17. *Under the homogeneous setting ($f_i = f$ for all i),* GreedMax *is guaranteed to find a partition $\hat{\pi}$ such that*

$$\min_{i=1,\ldots,m} f(A_i^{\hat{\pi}}) \geq \frac{1}{m} \max_{\pi \in \Pi} \min_{i=1,\ldots,m} f(A_i^{\pi}). \tag{7.4}$$

By assuming the homogeneity of the f_i's, we obtain a very simple $1/m$-approximation algorithm improving upon the state-of-the-art factor $1/(2m-1)$ [Khot and Ponnuswami, 2007]. Thanks to the lazy evaluation trick as described in [Minoux, 1978], Line 5 in Alg. 10 need not to recompute the marginal gain for every item in each round, leading GreedMax to scale to large data sets.

GreedSat: Though simple and effective in the homogeneous setting, GreedMax performs arbitrarily poorly under the heterogeneous setting. Consider the following example: $V = \{v_1, v_2\}$, $f_1(v_1) = 1, f_1(v_2) = 0, f_1(\{v_1, v_2\}) = 1, f_2(v_1) = 1+\epsilon, f_2(v_2) = 1$, $f_2(\{v_1, v_2\}) = 2 + \epsilon$. f_1 and f_2 are monotone submodular. The optimal partition is to assign v_1 to f_1 and v_2 to f_2 leading to a solution of value 1. However, GreedMax may assign v_1 to f_2 and v_2 to f_1 leading to a solution of value 0. Therefore, GreedMax performs arbitrarily poorly on this example.

To this end we provide another algorithm – "Saturate" Greedy (GreedSat, see Alg. 11). The key idea of GreedSat is to relax SFA to a much simpler problem – Submodular Wel-

Algorithm 11: GREEDSAT

1: Input: ϵ, $\{f_i\}_{i=1}^m$, m, V, α.
2: Let $\bar{F}^c(\pi) = \frac{1}{m}\sum_{i=1}^m \min\{f_i(A_i^\pi), c\}$.
3: Let $c_{\min} = 0$, $c_{\max} = \min_i f_i(V)$
4: **while** $c_{\max} - c_{\min} \geq \epsilon$ **do**
5: $c = \frac{1}{2}(c_{\max} + c_{\min})$
6: $\hat{\pi}^c \in \operatorname{argmax}_{\pi\in\Pi} \bar{F}^c(\pi)$ // solved by GREEDSWP (Alg 12)
7: **if** $\bar{F}^c(\hat{\pi}^c) < \alpha c$ **then**
8: $c_{\max} = c$
9: **else**
10: $c_{\min} = c$; $\hat{\pi} \leftarrow \hat{\pi}^c$
11: **end if**
12: **end while**
13: Output: $\hat{\pi}$.

fare (SWP), i.e., Problem 7.1 with $\lambda = 0$. Similar in flavor to the one proposed in [Krause et al., 2008a] GREEDSAT defines an intermediate objective $\bar{F}^c(\pi) = \sum_{i=1}^m f_i^c(A_i^\pi)$, where $f_i^c(A) = \frac{1}{m}\min\{f_i(A), c\}$ (Line 2). The parameter c controls the saturation in each block. It is easy to verify that f_i^c satisfies submodularity for each i. Unlike SFA, the combinatorial optimization problem $\max_{\pi\in\Pi} \bar{F}^c(\pi)$ (Line 6) is much easier and is an instance of SWP, which can be cast as maximization of submodular function under a partition matroid constraint (Problem 3). We solve Line 6 using the greedy algorithm as described in Alg. 4, which attains a constant 1/2-approximation [Fisher et al., 1978]. For completeness, we also describe the implementation detail of this algorithm in Alg. 12. Moreover the lazy evaluation trick also applies for Alg 12 enabling the wrapper algorithm GREEDSAT scalable to large data sets. One can also use a more computationally expensive multi-linear relaxation algorithm as given in [Vondrák, 2008] to solve Line 6 with a tight factor $\alpha = (1 - 1/e)$. Setting the input argu-

Algorithm 12: GreedSWP

Input: $\{f_i\}_{i=1}^m$, c, V

Initialize: $A_1 =, \ldots, = A_m = \emptyset$, and $R \leftarrow V$

while $R \neq \emptyset$ **do**

  for $i = 1, \ldots, m$ **do**

    $\delta_i = \max_{r \in R} \min\{f_i(A_i \cup r), c\} - \min\{f_i(A_i), c\}$

    $a_i \in \operatorname{argmax}_{r \in R} \min\{f_i(A_i \cup r), c\} - \min\{f_i(A_i), c\}$

  $j \in \operatorname{argmax}_i \delta(i)$

  $A_j \leftarrow A_j \cup \{a_j\}$

  $R \leftarrow R \setminus a_j$

Output $\hat{\pi}^c = (A_1, \ldots, A_m)$.

ment α as the approximation factor for Line 6, the essential idea of GreedSat is to perform a binary search over the parameter c to find the largest c^* such that the returned solution $\hat{\pi}^{c^*}$ for the instance of SWP satisfies $\bar{F}^{c^*}(\hat{\pi}^{c^*}) \geq \alpha c^*$. GreedSat terminates after solving $O(\log(\frac{\min_i f_i(V)}{\epsilon}))$ instances of SWP. Theorem 18 gives a bi-criterion optimality guarantee.

Theorem 18. *Given $\epsilon > 0$, $0 \leq \alpha \leq 1$ and any $0 < \delta < \alpha$,* GreedSat *finds a partition such that at least $\lceil m(\alpha - \delta) \rceil$ blocks receive utility at least $\frac{\delta}{1-\alpha+\delta}(\max_{\pi \in \Pi} \min_i f_i(A_i^\pi) - \epsilon)$.*

For any $0 < \delta < \alpha$ Theorem 18 ensures that the top $\lceil m(\alpha - \delta) \rceil$ valued blocks in the partition returned by GreedSat are $(\delta/(1-\alpha+\delta) - \epsilon)$-optimal. δ controls the trade-off between the number of top valued blocks to bound and the performance guarantee attained for these blocks. The smaller δ is, the more top blocks are bounded, but with a weaker guarantee. We set the input argument $\alpha = 1/2$ (or $\alpha = 1 - 1/e$) as the worst-case performance guarantee for solving SWP so that the above theoretical analysis follows. However, the worst-case is often achieved only by very contrived submodular functions. For the ones used in practice, the greedy algorithm often leads to near-optimal solution ([Krause et al., 2008a] and our own observations). Setting α as the actual performance guarantee for SWP (often very close to 1) can improve the empirical bound, and we, in practice, typically set $\alpha = 1$ to good effect.

Algorithm 13: MMax

Input: $\{f_i\}_{i=1}^m$, m, V, partition π^0.
Let $t = 0$.
repeat
  for $i = 1, \ldots, m$ **do**
    Pick a subgradient h_i at $A_i^{\pi^t}$ for f_i.
  end for
  $\pi^{t+1} \in \text{argmax}_{\pi \in \Pi} \min_i h_i(A_i^\pi)$
  $t = t + 1$;
until $\pi^t = \pi^{t-1}$
Output: $\hat{\pi} \in \text{argmax}_{\pi=\pi^1,\ldots,\pi^t} \min_i f_i(A_i^\pi)$.

MMax: In parallel to GreedSat, we also introduce a semi-gradient based approach for solving SFA under the heterogeneous setting. We call this algorithm minorization-maximization (MMax, see Alg. 13). Similar to the ones proposed in [Iyer et al., 2013b, Iyer and Bilmes, 2013], the idea is to iteratively maximize tight lower bounds of the submodular functions. Recall from Section 1.5.2 that submodular functions have tight modular lower bounds, which are related to the subdifferential $\partial_f(Y)$ of the submodular set function f at a set $Y \subseteq V$. For easy reference, we redefine the subdifferential below:

$$\partial_f(Y) = \{y \in \mathbb{R}^n : f(X) - y(X) \geq f(Y) - y(Y) \text{ for all } X \subseteq V\}. \tag{7.5}$$

For a vector $x \in \mathbb{R}^V$ and $X \subseteq V$, we write $x(X) = \sum_{j \in X} x(j)$. Denote a subgradient at Y by $h_Y \in \partial_f(Y)$, the extreme points of $\partial_f(Y)$ may be computed via a greedy algorithm: Let σ be a permutation of V that assigns the elements in Y to the first $|Y|$ positions ($\sigma(i) \in Y$ if and only if $i \leq |Y|$). Each such permutation defines a chain with elements $S_0^\sigma = \emptyset$, $S_i^\sigma = \{\sigma(1), \sigma(2), \ldots, \sigma(i)\}$, and $S_{|Y|}^\sigma = Y$. An extreme point h_Y^σ of $\partial_f(Y)$ has each entry as

$$h_Y^\sigma(\sigma(i)) = f(S_i^\sigma) - f(S_{i-1}^\sigma). \tag{7.6}$$

Defined as above, h^{σ}_{Y} forms a lower bound of f, tight at Y — i.e., $h^{\sigma}_{Y}(X) = \sum_{j \in X} h^{\sigma}_{Y}(j) \leq f(X), \forall X \subseteq V$ and $h^{\sigma}_{Y}(Y) = f(Y)$. The idea of MMax is to consider a modular lower bound tight at the set corresponding to each block of a partition. In other words, at iteration $t+1$, for each block i, we approximate f_i with its modular lower bound tight at $A_i^{\pi^t}$ and solve a modular version of Problem 1 (Line 7), which admits efficient approximation algorithms [Asadpour and Saberi, 2010]. MMax is initialized with a partition π^0, which is obtained by solving Problem 1, where each f_i is replaced with a simple modular function $f'_i(A) = \sum_{a \in A} f_i(a)$. The following worst-case bound holds:

Theorem 19. MMax *achieves a worst-case guarantee of*

$$O(\min_i \frac{1 + (|A_i^{\hat{\pi}}| - 1)(1 - \kappa_{f_i}(A_i^{\hat{\pi}}))}{|A_i^{\hat{\pi}}| \sqrt{m} \log^3 m}),$$

where $\hat{\pi} = (A_1^{\hat{\pi}}, \cdots, A_m^{\hat{\pi}})$ *is the partition obtained by the algorithm, and*

$$\kappa_f(A) = 1 - \min_{v \in V} \frac{f(v|A \setminus v)}{f(v)} \in [0, 1]$$

is the curvature of a submodular function f *at* $A \subseteq V$*.*

When each submodular function f_i is modular, i.e., $\kappa_{f_i}(A) = 0, \forall A \subseteq V, i$, the approximation guarantee of MMax becomes $O(\frac{1}{\sqrt{m} \log^3 m})$, which matches the performance of the approximation algorithm for the modular problem. When each f_i is fully curved, i.e., $\kappa_{f_i} = 1$, we still obtain a bounded guarantee of $O(\frac{1}{n\sqrt{m} \log^3 m})$. Theorem 19 suggests that the performance of MMax improves as the curvature κ_{f_i} of each objective f_i decreases. This is natural since MMax essentially uses the modular lower bounds as the proxy for each objective and optimizes with respect to the proxy functions. Lower the curvature of the objectives, the better the modular lower bounds approximate, hence better performance guarantee.

Since the modular version of SFA is also NP-hard and cannot be exactly solved in polynomial time, we cannot guarantee that successive iterations of MMax improves upon the overall objective. However we still obtain the following Theorem giving a bounded performance gap between the successive iterations.

Theorem 20. *Suppose modular version of SFA is solved with an approximation factor* $\alpha \leq 1$*, we have for each iteration* t *that*

$$\min_i f_i(A_i^{\pi_t}) \geq \alpha \min_i f_i(A_i^{\pi_{t-1}}). \tag{7.7}$$

7.1.2 Approximation Algorithms for SLB (Problem 7.2 with $\lambda = 0$*)*

In this section, we investigate the problem of submodular load balancing (SLB). It is a special case of Problem 7.2 with $\lambda = 0$. We first analyze the hardness of SLB. We then show a Lovász extension-based algorithm with a guarantee matching the problem's hardness. Lastly we describe a more efficient supergradient based algorithm.

Existing hardness for SLB is shown to be $o(\sqrt{n/\log n})$ [Svitkina and Fleischer, 2008]. However it is independent of m, and [Svitkina and Fleischer, 2008] assumes $m = \Theta(\sqrt{n/\log n})$ in their analysis. In most of the applications of SLB, we find that the parameter m is such that $m \ll n$ and can sometimes be treated as a constant w.r.t. n. To this end we offer a more general hardness analysis that is dependent directly on m.

Theorem 21. *For any* $\epsilon > 0$*, SLB cannot be approximated to a factor of* $(1-\epsilon)m$ *for any* $m = o(\sqrt{n/\log n})$ *with polynomial number of queries even under the homogeneous setting.*

Though the proof technique for Theorem 21 mostly carries over from [Svitkina and Fleischer, 2008], the result strictly generalizes the analysis in [Svitkina and Fleischer, 2008]. For any choice of $m = o(\sqrt{n/\log n})$ Theorem 21 implies that it is information theoretically hard to approximate SLB better than m even for the homogeneous setting. For the rest of the paper, we assume $m = o(\sqrt{n/\log n})$ for SLB, unless stated otherwise. It is worth pointing out that arbitrary partition $\pi \in \Pi$ already achieves the best approximation factor of m that one can hope for under the homogeneous setting. Denote π^* as the optimal partitioning for SLB, i.e., $\pi^* \in \operatorname{argmin}_{\pi \in \Pi} \max_i f(A_i^{\pi})$. This can be verified by considering the following:

$$\max_i f(A_i^{\pi}) \leq f(V) \leq \sum_{i=1}^{m} f(A_i^{\pi^*}) \leq m \max_i f(A_i^{\pi^*}). \tag{7.8}$$

Algorithm 14: LOVÁSZ ROUND

1: Input: $\{f_i\}_{i=1}^m$, $\{\tilde{f}_i\}_{i=1}^m$, m, V.
2: Solve for $\{x_i^*\}_{i=1}^m$ via convex relaxation.
3: Rounding: Let $A_1 =, \ldots, = A_m = \emptyset$.
4: **for** $j = 1, \ldots, n$ **do**
5: $\quad \hat{i} \in \text{argmax}_i\, x_i^*(j)$; $A_{\hat{i}} = A_{\hat{i}} \cup j$
6: **end for**
7: Output $\hat{\pi} = \{A_i\}_{i=1}^m$.

It is therefore theoretically interesting to consider only the heterogeneous setting.

Lovász Round: Next we propose a tight algorithm – LOVÁSZ ROUND (see Alg. 14) for the heterogeneous setting of SLB. The algorithm proceeds as follows: (1) apply the Lovász extension of submodular functions to relax SLB to a convex program, which is exactly solved to a fractional solution (Line 2); (2) map the fractional solution to a partition using the θ-rounding technique as proposed in [Iyer et al., 2014] (Line 3 - 6). The Lovász extension, which naturally connects a submodular function f with its convex relaxation $\tilde{f}$, is defined as follows: given any $x \in [0,1]^n$, we obtain a permutation σ_x by ordering its elements in non-increasing order, and thereby a chain of sets $S_0^{\sigma_x} \subset, \ldots, \subset S_n^{\sigma_x}$ with $S_j^{\sigma_x} = \{\sigma_x(1), \ldots, \sigma_x(j)\}$ for $j = 1, \ldots, n$. The Lovász extension $\tilde{f}$ for f is the weighted sum of the ordered entries of x:

$$\tilde{f}(x) = \sum_{j=1}^{n} x(\sigma_x(j))(f(S_j^{\sigma_x}) - f(S_{j-1}^{\sigma_x})). \tag{7.9}$$

Given the convexity of the $\tilde{f}_i$'s , SLB is relaxed to the following convex program:

$$\min_{x_1,\ldots,x_m \in [0,1]^n} \max_i \tilde{f}_i(x_i), \text{ s.t } \sum_{i=1}^{m} x_i(j) \geq 1, \text{ for } j = 1, \ldots, n \tag{7.10}$$

Denoting the optimal solution for Eqn 7.10 as $\{x_1^*, \ldots, x_m^*\}$, the θ-rounding step simply maps each item $j \in V$ to a block $\hat{i}$ such that $\hat{i} \in \text{argmax}_i\, x_i^*(j)$ (ties broken arbitrarily). The bound for LOVÁSZ ROUND is as follows:

Theorem 22. LOVÁSZ ROUND *is guaranteed to find a partition* $\hat{\pi} \in \Pi$ *such that*

$$\max_i f_i(A_i^{\hat{\pi}}) \leq m \min_{\pi \in \Pi} \max_i f_i(A_i^{\pi})$$

.

We remark that, to the best of our knowledge, LOVÁSZROUND is the first algorithm that is tight and that gives an approximation in terms of m for the heterogeneous setting.

Algorithm 15: MMIN

Input: $\{f_i\}_{i=1}^m$, m, V, partition π^0.
Let $t = 0$
repeat
  for $i = 1, \ldots, m$ **do**
    Pick a supergradient m_i at $A_i^{\pi^t}$ for f_i.
  end for
  $\pi^{t+1} \in \operatorname{argmin}_{\pi \in \Pi} \max_i m_i(A_i^{\pi})$
  $t = t + 1$;
until $\pi^t = \pi^{t-1}$
Output: π^t.

MMin: Similar to MMAX for SFA, we propose Majorization-Minimization (MMIN, see Alg. 15) for SLB. Here, we iteratively choose modular upper bounds, which, recall from Section 1.5.2, can be defined via superdifferentials $\partial^f(Y)$ of a submodular function at Y:

$$\partial^f(Y) = \{y \in \mathbb{R}^n : f(X) - y(X) \leq f(Y) - y(Y); \text{for all } X \subseteq V\}. \tag{7.11}$$

As shown in Section 1.5.2, there are specific supergradients that define the following two modular upper bounds (when referring to either one, we use m_X^f):

$$m_{X,1}^f(Y) \triangleq f(X) - \sum_{j \in X \setminus Y} f(j|X \backslash j) + \sum_{j \in Y \setminus X} f(j|\emptyset),$$

$$m^f_{X,2}(Y) \triangleq f(X) - \sum_{j \in X \setminus Y} f(j|V \setminus j) + \sum_{j \in Y \setminus X} f(j|X).$$

Then $m^f_{X,1}(Y) \geq f(Y)$ and $m^f_{X,2}(Y) \geq f(Y), \forall Y \subseteq V$ and $m^f_{X,1}(X) = m^f_{X,2}(X) = f(X)$. At iteration $t+1$, for each block i, MMin replaces f_i with a choice of its modular upper bound m_i tight at $A_i^{\pi^t}$ and solves a modular version of Problem 2 (Line 7), for which there exists an efficient LP relaxation based algorithm [Lenstra et al., 1990]. Similar to MMax, the initial partition π^0 is obtained by solving Problem 2, where each f_i is substituted with $f'_i(A) = \sum_{a \in A} f_i(a)$. The following worst-case bound holds:

Theorem 23. MMin *achieves a worst-case guarantee of* $(2 \max_i \frac{|A_i^{\pi^*}|}{1+(|A_i^{\pi^*}|-1)(1-\kappa_{f_i}(A_i^{\pi^*}))})$, *where* $\pi^* = (A_1^{\pi^*}, \cdots, A_m^{\pi^*})$ *denotes the optimal partition.*

Similar to MMax, we can show that MMin has bounded performance gaps in successive iterations.

Theorem 24. *Suppose the modular version of SLB can be solved with an approximation factor* $\alpha \geq 1$*, we have for each iteration* t *that*

$$\max_i f_i(A_i^{\pi_t}) \leq \alpha \max_i f_i(A_i^{\pi_{t-1}}). \tag{7.12}$$

7.2 General Submodular Partitioning (Problems 7.1 and 7.2 when $0 < \lambda < 1$*)*

In this section we study Problem 7.1 and Problem 7.2, in the most general case, i.e., $0 < \lambda < 1$. We use the proposed algorithms for the special cases of Problems 7.1 and 7.2 as the building blocks to design algorithms for the general scenarios ($0 < \lambda < 1$). We first propose a simple and generic scheme that provides performance guarantee in terms of λ for both problems. We then generalize the proposed GreedSat to obtain a more practically interesting algorithm for Problem 7.1. For Problem 7.2 we generalize Lovász Round to obtain a relaxation based algorithm.

Extremal Combination Scheme: First we describe the scheme that works for both problem 7.1 and 7.2. It naturally combines an algorithm for solving the worst-case problem

($\lambda = 0$) with an algorithm for solving the average case ($\lambda = 1$). We use Problem 7.1 as an example, but the same scheme easily works for Problem 7.2. Denote AlgWC as the algorithm for the worst-case problem (i.e. SFA), and AlgAC as the algorithm for the average case (i.e., SWP). The scheme is to first obtain a partition $\hat{\pi}_1$ by running AlgWC on the instance of Problem 7.1 with $\lambda = 0$ and a second partition $\hat{\pi}_2$ by running AlgAC with $\lambda = 1$. Then we output one of $\hat{\pi}_1$ and $\hat{\pi}_2$, with which the higher valuation for Problem 7.1 is achieved. We call this scheme CombSfaSwp. Suppose AlgWC solves the worst-case problem with a factor $\alpha \leq 1$ and AlgAC for the average case with $\beta \leq 1$. When applied to Problem 7.2 we refer to this scheme as CombSlbSmp ($\alpha \geq 1$ and $\beta \geq 1$). The following guarantee holds for both schemes:

Theorem 25. *For any* $\lambda \in (0, 1)$ CombSfaSwp *solves Problem 7.1 with a factor* $\max\{\frac{\beta\alpha}{\bar{\lambda}\beta+\alpha}, \lambda\beta\}$ *in the* heterogeneous *case, and* $\max\{\min\{\alpha, \frac{1}{m}\}, \frac{\beta\alpha}{\bar{\lambda}\beta+\alpha}, \lambda\beta\}$ *in the* homogeneous *case. Similarly,* CombSlbSmp *solves Problem 7.2 with a factor* $\min\{\frac{m\alpha}{m\bar{\lambda}+\lambda}, \beta(m\bar{\lambda}+\lambda)\}$ *in the* heterogeneous *case, and* $\min\{m, \frac{m\alpha}{m\bar{\lambda}+\lambda}, \beta(m\bar{\lambda}+\lambda)\}$ *in the* homogeneous *case.*

GeneralGreedSat: The drawback of CombSfaSwp and CombSlbSmp is that they do not explicitly exploit the trade-off between the average-case and worst-case in terms of λ. To obtain more practically interesting algorithms, we first give GeneralGreedSat (See Alg. 16) that generalizes GreedSat to solve Problem 7.1 for general λ. The key idea of GeneralGreedSat is again to relax Problem 7.1 to a simpler submodular welfare problem (SWP). Similar to GreedSat we define an intermediate objective:

$$\bar{F}^c_\lambda(\pi) = \frac{1}{m}\sum_{i=1}^{m} \min\{\bar{\lambda} f_i(A^\pi_i) + \lambda \frac{1}{m}\sum_{j=1}^{m} f_j(A^\pi_j), c\}. \tag{7.13}$$

It is easy to verify that the combinatorial optimization problem $\max_{\pi\in\Pi} \bar{F}^c_\lambda(\pi)$ (Line 6) can be formulated as the submodular welfare problem, for which we can solve efficiently with GreedSwp (see Alg. 12). Defining α as the optimality guarantee of the algorithm for solving Line 6 GeneralGreedSat solves Problem 7.1 with the following bounds:

Algorithm 16: GeneralGreedSat

Input: ϵ, $\{f_i\}_{i=1}^m$, m, V, λ, α.
Let $\bar{F}^c_\lambda(\pi) = \frac{1}{m}\sum_{i=1}^m \min\{\bar{\lambda} f_i(A_i^\pi) + \lambda\frac{1}{m}\sum_{j=1}^m f_j(A_j^\pi), c\}$.
Let $c_{\min} = 0$, $c_{\max} = \sum_{i=1}^m f_i(V)$
while $c_{\max} - c_{\min} \geq \epsilon$ **do**
  $c = \frac{1}{2}(c_{\max} + c_{\min})$
  $\hat{\pi}^c \in \operatorname{argmax}_{\pi\in\Pi} \bar{F}^c_\lambda(\pi)$ // solved by GreedSwp (Alg. 12)
  if $\bar{F}^c(\hat{\pi}^c) < \alpha c$ **then**
    $c_{\max} = c$
  else
    $c_{\min} = c$; $\hat{\pi} \leftarrow \hat{\pi}^c$
  end if
end while
Output: $\hat{\pi}$.

Theorem 26. *Given $\epsilon > 0$, $0 \leq \alpha \leq 1$, and $0 \leq \lambda \leq 1$,* GeneralGreedSat *finds a partition $\hat{\pi}$ that satisfies the following:*

$$\bar{\lambda} \min_i f_i(A_i^{\hat{\pi}}) + \lambda\frac{1}{m}\sum_{i=1}^m f_i(A_i^{\hat{\pi}}) \geq \lambda\alpha(OPT - \epsilon) \tag{7.14}$$

where $OPT = \max_{\pi\in\Pi} \bar{\lambda}\min_i f_i(A_i^\pi) + \lambda\frac{1}{m}\sum_{i=1}^m f_i(A_i^\pi)$.

Moreover, let $F_{\lambda,i}(\pi) = \bar{\lambda} f_i(A_i^\pi) + \lambda\frac{1}{m}\sum_{j=1}^m f_j(A_j^\pi)$. Given any $0 < \delta < \alpha$, there is a set $I \subseteq \{1,\ldots,m\}$ such that $|I| \geq \lceil m(\alpha - \delta)\rceil$ and

$$F_{i,\lambda}(\hat{\pi}) \geq \max\{\frac{\delta}{1-\alpha+\delta}, \lambda\alpha\}(OPT - \epsilon), \forall i \in I. \tag{7.15}$$

Eqn 7.15 in Theorem 26 reduces to Theorem 18 when $\lambda = 0$, i.e., it recovers the bi-criterion guarantee in Theorem 18 for the worst-case scenario ($\lambda = 0$). Eqn 7.14 in Theorem 26 implies that α-approximation for the average-case objective can almost be recovered

Algorithm 17: GENERALLOVÁSZ ROUND

1: Input: $\{f_i\}_{i=1}^m$, $\{\tilde{f}_i\}_{i=1}^m$, λ, m, V.

2: Solve

$$\min_{x_1,\ldots,x_m\in[0,1]^n} \max_i \bar{\lambda}\tilde{f}_i(x_i) + \lambda\frac{1}{m}\sum_{j=1}^m \tilde{f}_j(x_j), \text{ s.t } \sum_{i=1}^m x_i(j) \geq 1, \text{ for } j = 1,\ldots,n \quad (7.16)$$

for $\{x_i^*\}_{i=1}^m$ via convex relaxation.

3: Rounding: Let $A_1 =, \ldots, = A_m = \emptyset$.

4: **for** $j = 1, \ldots, n$ **do**

5: $\quad \hat{i} \in \text{argmax}_i\, x_i^*(j)$; $A_{\hat{i}} = A_{\hat{i}} \cup j$

6: **end for**

7: Output $\hat{\pi} = \{A_i\}_{i=1}^m$.

by GENERALGREEDSAT if $\lambda = 1$. Moreover Theorem 26 shows that the guarantee of GENERALGREEDSAT improves as λ increases suggesting that Problem 7.1 becomes easier as the mixed objective weights more on the average-case objective. We also point out that the approximation guarantee of GENERALGREEDSAT smoothly interpolates the two extreme cases in terms of λ.

GeneralLovász Round: Next we focus on designing practically more interesting algorithms for Problem 7.2 with general λ. In particular we generalize LOVÁSZ ROUND leading to GENERALLOVÁSZ ROUND as shown in Alg. 17. Sharing the same idea with LOVÁSZ ROUND, GENERALLOVÁSZ ROUND first relaxes Problem 7.2 as a convex program (defined in Eqn 7.16) using the Lovász extension of each submodular objective. Given the fractional solution to the convex program $\{x_i^*\}_{i=1}^m$, the algorithm then rounds it to a partition using the θ-rounding technique (Line 3- 6). Note the rounding technique used for GENERALLOVÁSZ ROUND is the same as in LOVÁSZ ROUND. The following Theorem holds:

Theorem 27. GeneralLovász Round *is guaranteed to find a partition* $\hat{\pi} \in \Pi$ *such that*

$$\max_i \bar{\lambda} f_i(A_i^{\hat{\pi}}) + \lambda \frac{1}{m} \sum_{j=1}^{m} f_j(A_j^{\hat{\pi}}) \leq m \min_{\pi \in \Pi} \max_i \bar{\lambda} f_i(A_i^{\pi}) + \lambda \frac{1}{m} \sum_{j=1}^{m} f_j(A_j^{\pi}). \tag{7.17}$$

Theorem 27 generalizes Theorem 22 when $\lambda = 0$. Moreover we achieve a factor of m by GeneralLovász Round for any λ. Though the approximation guarantee is independent of λ the algorithm naturally exploits the trade-off between the worst-case and average-case objectives in terms of λ. The drawback of GeneralLovász Round is that it requires high order polynomial queries of the Lovász extension of the submodular objectives, hence is not computationally feasible for even medium sized tasks. Moreover, if we restrict ourselves to the homogeneous setting (f_i's are identical), it is easy to verify that arbitrary partitioning already achieves a guarantee of m while Problem 7.2, in general, cannot be approximated better than m as shown in Theorem 21.

GeneralGreedMin: In this case, we should resort to intuitive heuristics that are scalable to large-scale applications to solve Problem 7.2 with general λ. To this end we design a greedy heuristic called GeneralGreedMin (see Alg. 18).

The algorithm first solves a constrained submodular maximization on f to obtain a set S_{seed} of m seeds. Since f is submodular, maximizing f always leads to a set of diverse seeds, where the diversity is measured by the objective f. We initialize each block A_i with one seed from S_{seed}. Defining k as the number of items that have already been assigned. The main algorithm consists of two phases. In the first phase ($k \leq (1-\lambda)|V|$), we, for each iteration, assign the item that has the smallest marginal gain to the block whose valuation is the smallest. Since the functions are all monotone, any additions to a block can (if anything) only increase its value. Such procedure inherently minimizes the worst-case objective, since it chooses the minimum valuation block to add to in order to keep the maximum valuation block from growing further. In the second phase ($k > \lambda|V|$), we assign an item such that its marginal gain is the smallest among all remaining items and all blocks. The greedy procedure in this phase, on the hand, is suitable for minimizing the average-case objective, since it, in each iteration, assigns an item so that the valuation of the average-case objective increases

Algorithm 18: GeneralGreedMin

1: Input: f, m, V, $0 \leq \lambda \leq 1$;
2: Solve $S_{\text{seed}} \in \text{argmax}_{S \subseteq V; |S|=m} f(S)$ for m seeds with $S_{\text{seed}} = \{s_1, \ldots, s_m\}$. Initialize each block i by the seeds as $A_i \leftarrow \{s_i\}, \forall i$. Initialize a counter as $k = m$ and $R = V \setminus S_{\text{seed}}$.
3: **while** $R \neq \emptyset$ **do**
4: **if** $k \leq (1 - \lambda)|V|$ **then**
5: $j^* \in \text{argmin}_j f(A_j)$; $a^* \in \min_{a \in R} f(a|A_{j^*})$
6: $A_{j^*} \leftarrow A_{j^*} \cup a^*$; $R \leftarrow R \setminus a^*$
7: **else**
8: **for** $i = 1, \ldots, m$ **do**
9: $a_i^* \in \text{argmin}_{a \in R} f(a|A_i)$
10: **end for**
11: $j^* \in \text{argmin}_{i=1,\ldots,m} f(a_i^*|A_i)$; $A_{j^*} \leftarrow A_{j^*} \cup a^*$; $R \leftarrow R \setminus a_{j^*}^*$
12: **end if**
13: $k = k + 1$;
14: **end while**
15: Output $\{A_i\}_{i=1}^m$.

the least. The trade-off between the worse-case and the average-case objectives is controlled by λ, which is used as the input argument to the algorithm. In particular, λ controls the fraction of the iterations in the algorithm to optimize the average-case objective. When $\lambda = 1$, the algorithm solely focuses on the average-case objective, while only the worst-case objective is minimized if $\lambda = 0$.

In general GeneralGreedMin requires $O(m|V|^2)$ function valuations, which may still be computationally difficult for large-scale applications. In practice, one can relax the condition in Line 5 and 9. Instead of searching among all items in R, one can, in each round, randomly select a subset $\hat{R} \subseteq R$ and choose an item with the smallest marginal gain from

only the subset $\hat{R}$. The resultant computational complexity is reduced to $O(m|\hat{R}||V|)$ function valuations. Empirically we observe that GeneralGreedMin can be sped up more than 100 times by this trick without much performance loss.

7.3 Applications

7.3.1 Applications of Problem 7.1

Distributed statistical training: An important machine learning application is distributed training of statistical models. As data set sizes grow, the need for statistical training procedures tolerant of the distributed data partitioning becomes more important. Existing schemes are often developed and performed assuming data samples are distributed to their computational clients in an arbitrary or random fashion. As an alternate strategy, if the data is intelligently partitioned such that each block of samples can itself lead to a good approximate solution, a consensus amongst the distributed results could be reached more quickly than when under a poor partitioning. As shown in Chapter 4, submodular functions can in fact express the value of a subset of training data for certain machine learning risk functions. Using these functions within Problem 7.1, one can expect a partitioning (by formulating the problem as an instance of Problem 7.1, $\lambda \approx 0$) where each block is a good representative of the entire set, thereby achieving faster convergence in distributed settings. We defer the readers to Chapter 8 for a concrete case study on this application.

7.3.2 Applications of Problem 7.2

Data clustering and image segmentation: Submodular functions naturally capture notions of interacting cooperative costs and homogeneity and thus are useful for clustering and image segmentation [Narasimhan et al., 2005, Boykov and Jolly, 2001, Kolmogorov and Zabih, 2004, Kohli et al., 2013]. While the average case instance (Problem 7.2 with $\lambda = 1$) has been used before, a more worst-case variant (i.e., Problem 7.2 with $\lambda \approx 0$) is useful to produce balanced clusterings (i.e., the submodular valuations of all the blocks should be

similar to each other). Problem 7.2 also addresses a problem in image segmentation, namely how to use only submodular functions (which are instances of pseudo-Boolean functions) for multi-label (i.e., non-Boolean) image segmentation. Problem 7.2 addresses this problem by allowing each segment j to have its own submodular function f_j, and the objective measures the homogeneity $f_j(A_j^\pi)$ of segment j based on the image pixels A_j^π assigned to it. Moreover, by combining the average case and the worst case objectives, one can achieve a tradeoff between the two. Empirically, we evaluate our algorithms on unsupervised image segmentation (Section 6.5) and find that it outperforms other clustering methods including k-means, k-medoids, graph cut, and spectral clustering.

Computational load balancing: Submodularity also accurately represents computational costs in distributed systems, as shown in [Li et al., 2015]. They consider a problem of text data partitioning for balancing memory demands. Given a large collection of documents $V = \{v_1, \ldots, v_n\}$, the goal is to distribute the documents into m machines such that the memory requirements across the machines are balanced and minimized. Each document $v \in V$ consists a set of keys, and let $U = \{u_1, \ldots, u_k\}$ be the set of all possible keys. $|U|$ can be in the order of billions (e.g., the set of all unigrams, bigrams, and trigrams). Let $N(v_i) \subseteq U$ be the set of keys contained by the document v_i. Given a partition $\pi = (A_1^\pi, \ldots, A_m^\pi)$ of the documents V, the number of unique keys associated with the collection A_i^π on machine i is expressed as

$$f_{\text{sc}}(A_i^\pi) = |\cup_{v \in A_i^\pi} N(v)|, \tag{7.18}$$

where f_{sc} is the set cover function. A hard constraint for a partition $\{X_1, \ldots, X_m\}$ to satisfy is that the number of unique keys on each machine has to be small enough so that they can be cached into each machine's memory. Since the memory needed to cache the keys on machine i is proportional to $|\cup_{v \in X_i} N(v)|$, a feasible partition of the documents satisfying the memory requirement can, therefore, be found by solving the following:

$$\min_{\pi \in \Pi} \max_{i=1,\ldots,m} f_{\text{sc}}(A_i^\pi), \tag{7.19}$$

which is an instance of Problem 7.2 for $\lambda = 0$ with f_{sc} as the objective function.

7.4 Experimental Results

7.4.1 Simulations on Synthetic Data Sets

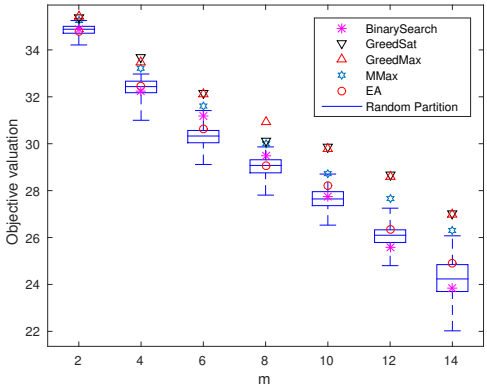
In this section we evaluate separately on four different cases: Problem 7.1 with $\lambda = 0$ (SFA), Problem 7.2 with $\lambda = 0$ (SLB), Problem 7.1 with $0 < \lambda < 1$, and Problem 7.2 with $0 < \lambda < 1$. Since some of the algorithms, such as the Ellipsoidal Approximations [Goemans et al., 2009] and Lovász relaxation algorithms, are computationally intensive, we restrict ourselves to only 40 data instances, i.e., $|V| = 40$. For simplicity we only evaluate on the homogeneous setting (f_i's are identical). For each case we test with two types of submodular functions: facility location function, and the set cover function. The facility location function is defined as follows:

$$f_{\text{fac}}(A) = \sum_{v \in V} \max_{a \in A} s_{v,a}, \tag{7.20}$$

where $s_{v,a}$ is the similarity between item v and a and is symmetric, i.e., $s_{v,a} = s_{a,v}$ for any pair of v and a. We define f_{fac} on a complete similarity graph with each edge weight $s_{v,a}$ sampled uniformly and independently from $[0, 1]$. The set cover function is defined in Eqn 7.18. In the experiments we choose $|U| = 40$ and define f_{sc} over a bipartite graph between V and U. An edge between an item $v \in V$ and an object $u \in U$ is defined independently with probability $p = 0.2$.

Problem 7.1 For $\lambda = 0$, i.e., SFA, we compare among 6 algorithms: GreedMax, GreedSat, MMax, Balanced Partition (BP), Ellipsoid Approximation (EA) [Goemans et al., 2009], and Binary Search algorithm (BS) [Khot and Ponnuswami, 2007]. Balanced Partition method simply partitions the ground set V into m blocks such that the size of each block is balanced and is either $\lceil \frac{|V|}{m} \rceil$ or $\lfloor \frac{|V|}{m} \rfloor$. We run 100 randomly generated instances of the balanced partition method. GreedSat is implemented with the choice of the hyperparameter $\alpha = 1$. We compare the performance of these algorithms in Figure 7.1a and 7.1b, where we vary the number of blocks m from 2 to 14. The three proposed algorithms (GreedMax,

(a) Problem 7.1 on f_{fac} with $\lambda = 0$



(b) Problem 7.1 on f_{sc} with $\lambda = 0$

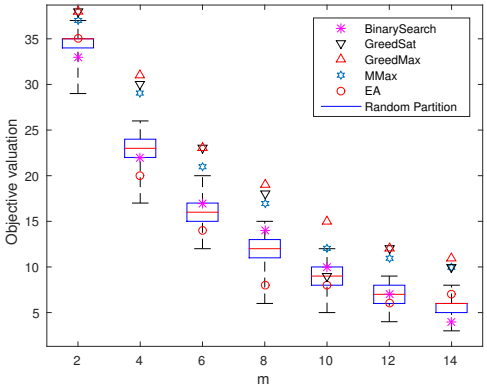


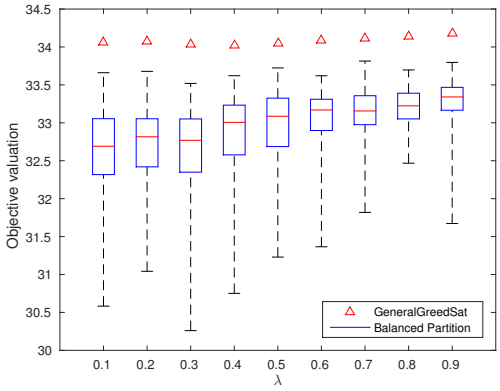
Figure 7.1: Synthetic experiments on Problem 7.1 with $\lambda = 0$ on facility location function (a) and set cover function (b).

GreedSat, and MMax) significantly and consistently outperform all baseline methods for both f_{fac} and f_{sc}. Among the proposed algorithms we observe that GreedMax, in general, yields the superior performance. Given the empirical success, computational efficiency, and tight theoretical guarantee, we suggest GreedMax as the first choice of algorithm to solve SFA under the homogeneous setting.

Next we evaluate Problem 7.1 with general $0 < \lambda < 1$. Baseline algorithms for SFA such as Ellipsoidal Approximations, Binary Search do not apply to the mixed scenario. Similarly the proposed algorithms such as GreedMax, MMax do not simply generalize to this scenario. We therefore only compare GeneralGreedSat with the Balanced Partition as a baseline. The results are summarized in Figure 7.2a and 7.2b. We observe that General-GreedSat consistently and significantly outperform even the best of 100 instances of the baseline method for all cases of λ.

Problem 7.2 For $\lambda = 0$, i.e., SLB, we compare among 5 algorithms: Lovász Round, MMin, GeneralGreedMin, Ellipsoid Approximation (EA) [Goemans et al., 2009], and

(a) Problem 7.1 on f_{fac} with varying λ



(b) Problem 7.1 on f_{sc} with varying λ

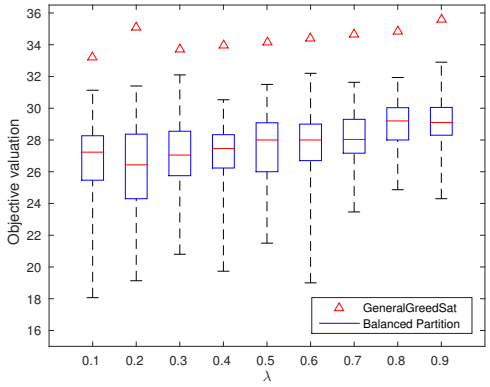


Figure 7.2: Problem 7.1 with general $0 < \lambda < 1$ on facility location function (a) and set cover function (b).

Balanced Partition [Svitkina and Fleischer, 2011]. We implement GeneralGreedMin with the input argument $\lambda = 0$. We also run 100 randomly generated instances of the Balanced Partition method as a baseline. We show the results in Figure 7.3a and 7.3b. Among all five algorithms MMin and GeneralGreedMin, in general, perform the best. Between MMin and GeneralGreedMin we observe that GeneralGreedMin performs marginally better, especially on f_{sc}. The computationally intensive algorithms, such as Ellipsoid Approximation and Lovász Round, do not perform well, though they carry better worst-case approximation factors for the heterogeneous setting.

Lastly we evaluate Problem 7.2 with general $0 < \lambda < 1$. Since MMin and Ellipsoid Approximation do not apply for the mixed scenario, we test only on GeneralLovász Round, GeneralGreedMin, and Balanced Partition. Again we test on 100 instances of randomly generated balanced partitions. We vary λ in this experiment. The results are shown in Figure 7.4a and 7.4b. The best performance is consistently achieved by GeneralGreedMin.

(a) Problem 7.2 on f_{fac} with $\lambda = 0$ (b) Problem 7.2 on f_{sc} with $\lambda = 0$

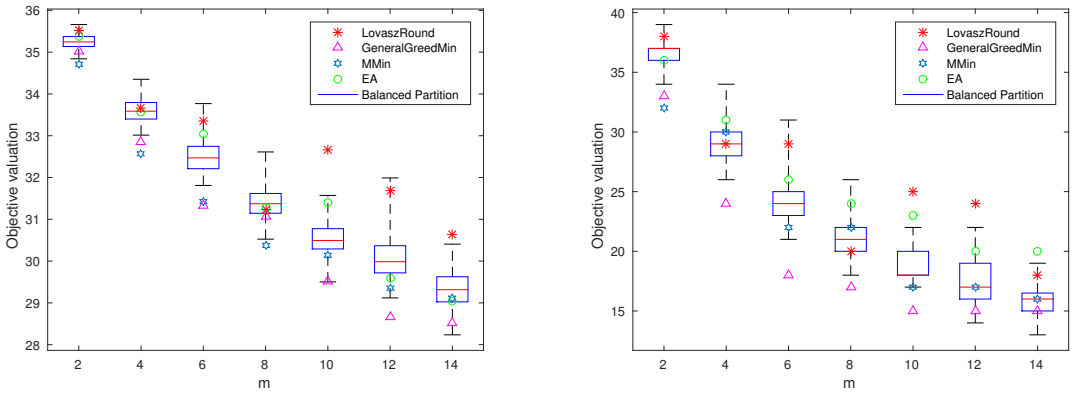


Figure 7.3: Synthetic experiments on Problem 7.2 with $\lambda = 0$ on facility location function (a) and set cover function (b).

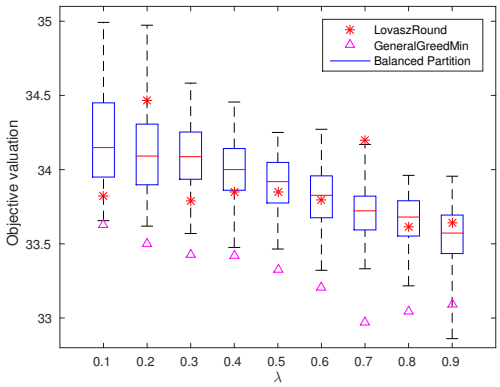
7.4.2 Unsupervised Image Segmentation

Lastly we test the efficacy of Problem 7.2 on the task of unsupervised image segmentation. We evaluate on the Grab-Cut data set, which consists of 30 color images. Each image has ground truth foreground/background labels. By "unsupervised", we mean that no labeled data at any time in supervised or semi-supervised training, nor any kind of interactive segmentation, was used in forming or optimizing the objective. In our experiments, the image segmentation task is solved as unsupervised clustering of the pixels, where the goal is to obtain a partitioning of the pixels such that the majority of the pixels in each block share either the same foreground or the background labels.

Let V be the ground set of pixels of an image, π be an m-partition of the image, and $\{y_v\}_{v \in V}$ as the pixel-wise ground truth label ($y_v = \{0, 1\}$ with 0 being background and 1 being foreground). We measure the performance of the partition π in the following two steps:

1. For each block i, predict $\hat{y}_v$ for all the pixels $v \in A_i^\pi$ in the block as either 0 or 1 having larger intersection with the ground truth labels, i.e., predict $\hat{y}_v = 1, \forall v \in A_i^\pi$, if

(a) Problem 7.2 on f_{fac} with varying λ



(b) Problem 7.2 on f_{sc} with varying λ

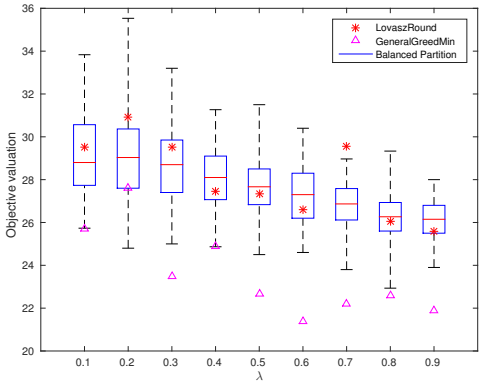


Figure 7.4: Problem 7.2 with general $0 < \lambda < 1$ on facility location function (a) and set cover function (b).

$\sum_{v \in A_i^\pi} \mathbb{1}\{y_v = 1\} \geq \sum_{v \in A_i^\pi} \mathbb{1}\{y_v = 0\}$, and predict $\hat{y}_v = 0, \forall v \in A_i^\pi$ otherwise.

2. Report the performance of the partition π as the F-measure of the predicted labels $\{\hat{y}_v\}_{v \in V}$ relative to the ground truth label $\{y_v\}_{v \in V}$.

In the experiment we first preprocess the data by downsampling each image by a factor 0.25 for testing efficiency. We represent each pixel v as 5-dimensional features $x_v \in \mathbb{R}^5$, including the RGB values and pixel positions. We normalize each feature within $[0, 1]$. To obtain a segmentation of each image we solve an instance of Problem 7.2 ($0 < \lambda < 1$) under the homogeneous setting using GeneralGreedMin (Alg. 18). We use the facility location function f_{fac} as the objective for Problem 7.2. The similarity $s_{v,a}$ between the pixels v and a is computed as $s_{v,a} = C - \|x_v - x_a\|_2$ with $C = \max_{v,v' \in V} \|x_v - x'_v\|_2$ being the maximum pairwise Euclidean distance. Since the facility location function f_{fac} is defined on a pairwise similarity graph, which requires $O(|V|^2)$ memory complexity. It becomes computationally infeasible for medium sized images. Fortunately a facility location function that is defined on a sparse k-nearest neighbor similarity graph performs just as well with k being very sparse

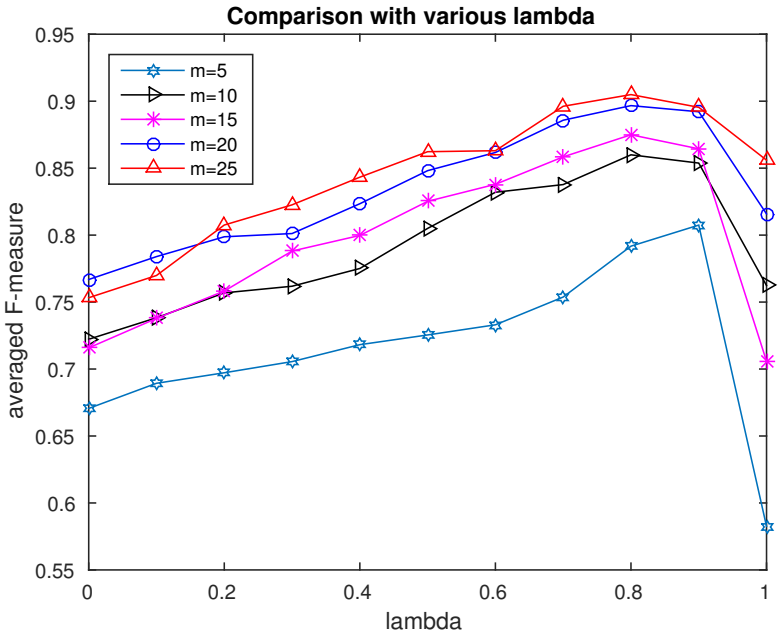


Figure 7.5: Comparison of our method with varying λ.

as discussed in Chapter 6. In the experiment, we instantiate f_{fac} by a sparse 10-nearest neighbor sparse graph, where each item v is connected only to its 10 closest neighbors.

We test against the following unsupervised methods as baselines in the experiment:

1. k-means,

2. k-medoids,

3. graph cuts [Boykov and Kolmogorov, 2004] ,

4. spectral clustering [Von Luxburg, 2007].

We use the RBF kernel sparse similarity matrix as the input for spectral clustering. The sparsity of the similarity matrix is k and the width parameter of the RBF kernel σ. We test with various choices of σ and k and find that the setting of $\sigma = 1$ and $k = 20$ performs the

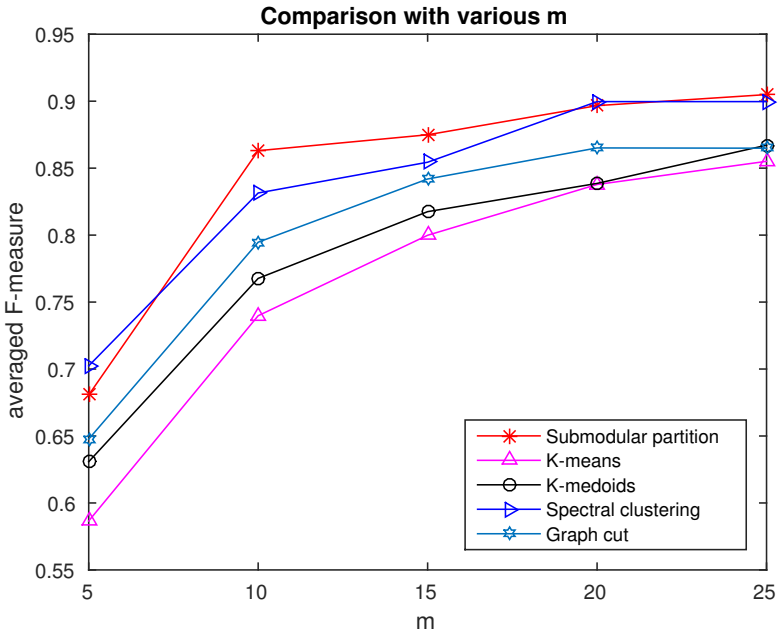


Figure 7.6: Comparison of our method against baseline methods with varying m.

best, with which we report the results. For graph cuts, we use the MATLAB implementation [Bagon, 2006], which has a smoothness parameter α. We tune $\alpha = 0.3$ to achieve the best performance and report the result of graph cuts using this choice.

The proposed image segmentation method involves a hyperparameter λ, which controls the trade-off between the worst-case objective and the average-case objective. First we examine how the performance of our method varies with different choices of λ in Figure 7.5. The performance is measured as the averaged F-measure of a partitioning method over all images in the data set. Interestingly we observe that the performance smoothly varies as λ increases from 0 to 1. In particular the best performance is achieved when λ is within the range $[0.7, 0.9]$. It suggests that using only the worst-case or the average-case objective does not suffice for the unsupervised image segmentation / clustering task, and an improved result is achieved by mixing these two extreme cases. In the subsequent experiments we show only the result of our method with $\lambda = 0.2$.

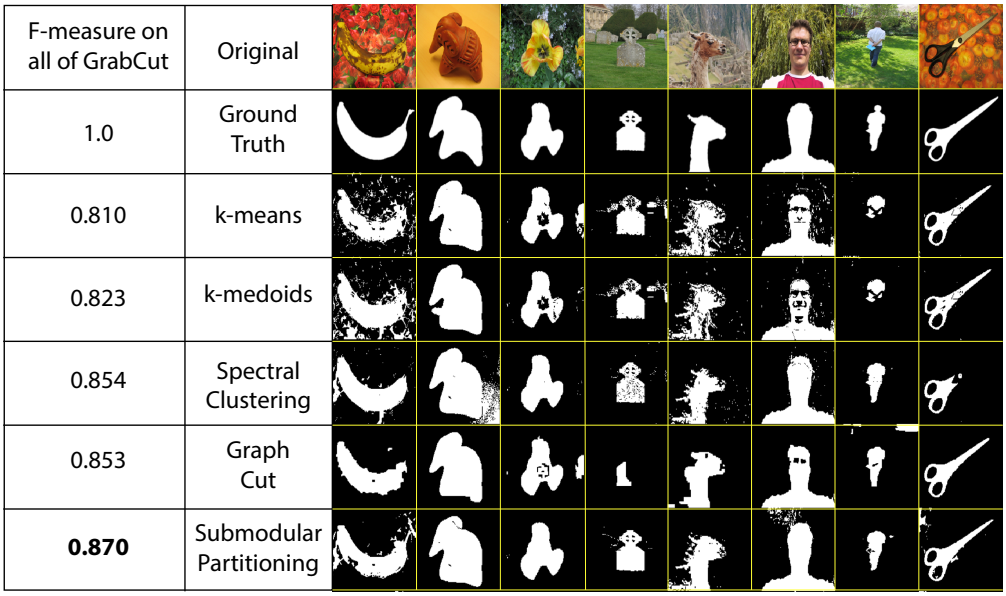


Figure 7.7: Unsupervised image segmentation (right: some examples).

Next we compare the proposed approach with baseline methods on various m in Figure 7.6. In general, each method improves as m increases. Submodular partitioning method performs the best on almost all cases of m. Lastly we show in Figure 7.7 example segmentation results on several example images as well as averaged F-measure in the case of $m = 15$. We observe that submodular partitioning, in general, leads to less noisy and more coherent segmentation in comparison to the baselines.

7.5 Discussion

In this chapter, we considered two novel mixed robust/average-case submodular partitioning problems, which generalize four well-known problems: submodular fair allocation (SFA), submodular load balancing (SLB), submodular welfare problem (SWP), and submodular multiway partition (SMP). While the average case problems, i.e., SWP and SMP, admit efficient and tight algorithms, existing approaches for the worst case problems, i.e., SFA and SLB, are, in general, not scalable. We bridge this gap by providing several new algorithms

that not only scale to large data sets but that also achieve comparable theoretical guarantees. Moreover we provide a number of efficient frameworks for solving the general mixed robust/average-case submodular partitioning problems. We also demonstrate that submodular partitioning is applicable in a number of machine learning problems involving distributed optimization, computational load balancing, and unsupervised image segmentation. Lastly we empirically show the effectiveness of the proposed algorithms on these machine learning tasks.

Future work will concentrate on proving hardness bound for SFA and correspondingly Problem 7.1 with general λ. Though GreedMax improves upon the state-of-the-art algorithm by [Khot and Ponnuswami, 2007] for SFA under the homogeneous setting, we hypothesize that a tighter algorithm with a constant factor guarantee may be designed for this scenario. Other variants of the submodular partitioning should also be interesting to analyze. In particular, it is worth considering the problem with size constraints in addition to the partition constraint. Another direction would be to consider the scenario of allowing each item to be included in multiple blocks. This case is natural in file systems where each data file need to be duplicated and stored in multiple machines. Aside from the theoretical investigation, it is also interesting to come up with more practical applications that may be naturally formulated in terms of submodular partitioning.

Chapter 8

CASE STUDY I: DISTRIBUTED MACHINE LEARNING VIA INTELLIGENT DATA PARTITIONING

In this chapter we perform a case study of the submodular partitioning problem (Problem 7.1) on training data partitioning for parallel learning of statistical models. Big data presents significant computational challenges to machine learning since, while big data is still getting bigger, it is expected that we are nearing the end of Moore's law [Thompson and Parthasarathy, 2006], and single threaded computing speed has unfortunately not significantly improved since about 2003. It is hence imperative to develop efficient and scalable methods for large scale training of statistical models. Parallel and distributed computing approaches are natural for this challenge.

Since machine learning procedures are performed over sets of data, one simple way to achieve parallelism is to split the data into chunks each of which resides on a compute node. This is the idea behind many parallel learning approaches such as ADMM [Boyd et al., 2011] and distributed neural network training [Povey et al., 2014], to name only a few. Such parallel schemes are often performed where the data samples are distributed to their compute nodes in an arbitrary or random fashion. However, there has apparently been very little work on how to intelligently split the data to ensure that the resultant model can be learned in a more efficient manner.

One way to approach this problem is to consider a class of "utility" functions on the training data. Given a set $V = \{v_1, \ldots, v_n\}$ of training data items, suppose that we have a set function $f : 2^V \rightarrow \mathbb{R}_+$ that measures the utility in subsets of the data set V. That is, given any $A \subseteq V$, $f(A)$ measures the utility of the training data subset A for producing a good resulting trained model. Given a parallel training scheme (e.g., ADMM) with m

compute nodes, we consider an m-partition $\pi = (A_1^\pi, A_2^\pi, \ldots, A_m^\pi)$ (i.e., $\cup_i A_i^\pi = V$ and $\forall i \neq j, A_i^\pi \cap A_j^\pi = \emptyset$) of the entire training data V, where we send the i^{th} block A_i^π of the partition π to the i^{th} compute node. If each compute node i has a block of the data A_i^π that is non-representative of the utility of the whole (i.e., $f(A_i^m) \ll f(V)$), the parallel learning algorithm, at each iteration, might have the compute nodes deduce models that are widely different from each other. Any subsequent aggregation of the models could then result in a joint model that is non-representative of the whole, especially in a non-convex case like deep neural network models. On the other hand, suppose that an intelligent partition π of the training data V is achieved such that each block A_i^π is highly representative of the whole (i.e., $f(A_i^m) \approx f(V), \forall i$). In this case, the models deduced at the compute nodes will tend to be close to a model trained on the entire data, and any aggregation of the resulting models (say via ADMM) will likely be better. An intelligent data partition, in fact, may have a positive effect in two ways:

1. It can lead to a better final solution (in the non-convex case),

2. Faster convergence may be achieved even in the convex case thanks to the fact that any "oscillations" between a distributed stage (where the compute nodes are operating on local data) and an aggregation stage (where some form of model average is computed) could be dampened and hence reduced.

8.1 Problem Formulation

In this chapter, based on the above intuition, we mathematically describe this goal as obtaining an m-partition of the data set V such that the worst-case or the average-case utility among all blocks in the partition is maximized. More precisely, this can be formulated as follows:

$$\max_{\pi \in \Pi} \Big[(1-\lambda) \min_i f_i(A_i^\pi) + \frac{\lambda}{m} \sum_{j=1}^{m} f_j(A_j^\pi) \Big], \tag{8.1}$$

where the set of sets $\pi = (A_1^\pi, A_2^\pi, \cdots, A_m^\pi)$ forms a partition of the finite set V, Π refers to the set of all partitions of V into m blocks, and f_i models the utility score assigned to i^{th} block A_i^π of the partition. The first term $\min_i f_i(A_i^\pi)$ valuates the worst-case utility among the blocks in the partition π, while the second term $\frac{1}{m}\sum_{i=1}^m f_i(A_i^\pi)$ measures the averaged utility of the partition. The parameter λ controls the objective: $\lambda = 1$ is the average case, $\lambda = 0$ is the robust case, and $0 < \lambda < 1$ is a mixed case.

In general, Problem 8.1 is hopelessly intractable, even to approximate. However, as we have shown in Chapter 7, the problem becomes tractable and can be approximately solved by efficient and scalable algorithms when we assume f_i's to be monotone submodular. These assumptions, moreover, allow us to retain the naturalness and applicability of Problem 8.1 to the goal at hand. Submodular functions are an ideal class of functions for modeling information over data sets. For example, we demonstrated in Chapter 4 that the utility functions of data subsets for training certain machine learning classifiers can be derived as submodular functions. If f is selected as the class of submodular functions that appropriately model the notion of utility for a given machine learning setting (which could be different depending, say, on what form of training one is doing), solving the homogeneous instance of Problem 8.1 with f as the objective, then, addresses a core goal in modern machine leanring on big data sets, namely how to intelligently partition and distribute training data to multiple compute nodes.

It should be noted that a random partition might have a high probability of performing well, and might have exponentially small probability of producing a partition that is worst case. On the other hand, there are also quite likely a small number of partitions that perform exceedingly well and a random partition also has exponentially small probability of landing on one of these high quality partitions. Our quest is to develop methods that increase the likelihood that we can discover one of these rare but high performing partitions.

Chapter 7 has already fully described all algorithms, theorems demonstrating mathematical guarantees, and related work regarding both Problem 8.1 and also its dual companion problem that is useful for other applications. In this chapter, we concentrate on the afore-

mentioned application. We omit the discussion of the previous work on Problem 8.1, which has been described in Chapter 7. In general, much of the previous work focuses on designing theoretically tighter algorithms that are not as scalable. Moreover, in the below we concentrate on the robust $\lambda = 0$ case under the homogeneous setting (f_i's are identical to each other) for now, although we believe that more experiments with the right submodular functions will show some utility in the $0 < \lambda < 1$ case as well.

8.2 Approximation Algorithms for Problem 8.1 with $\lambda = 0$

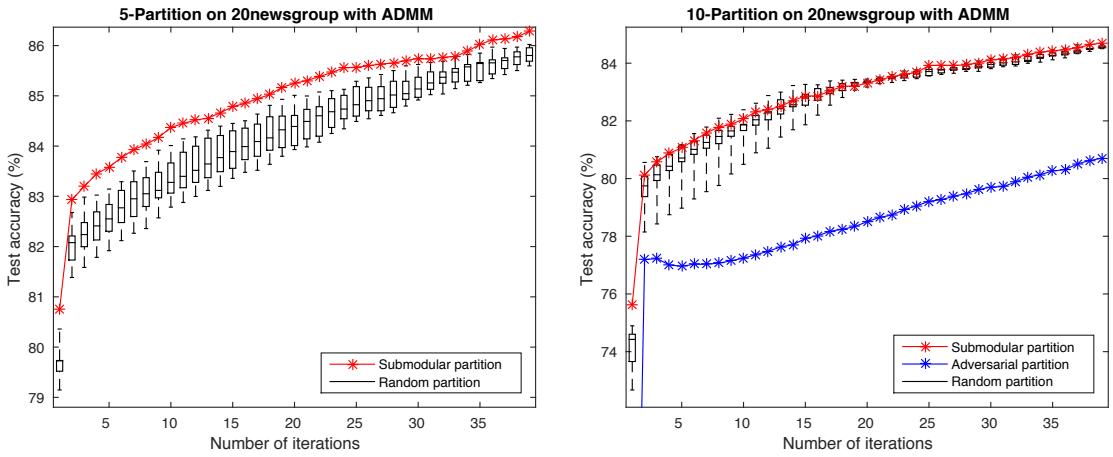
The homogeneous instance of Problem 8.1 with $\lambda = 0$ can be equivalently stated as follows:

$$\max_{\pi \in \Pi} \min_{i} f(A_i^{\pi}). \tag{8.2}$$

We gave an algorithm with a greedy flavor (GreedMax, see Alg. 10) in Chapter 7 for this problem. The key idea of GreedMax is to greedily add an item with the maximum marginal gain to the block whose current solution is minimum. By assuming the homogeneity of the f_i's, GreedMax attains a $1/m$-approximation on this problem. Moreover, GreedMax is scalable to large data sets thanks to the lazy evaluation trick as described in [Minoux, 1978].

Another variant of the greedy algorithm (GreedSat, see Alg. 11) can also approximately solve Problem 8.1 with a bicriterion guarantee. However, this algorithm was designed for a more general case–heterogeneous setting. It does not fully exploit the homogeneous structure in our problem formulation here. Moreover, it requires much more computational cost than GreedMax. Hence, we believe that GreedMax should better suit the application considered in this Chapter.

In the context of parallel machine learning, obtaining a partition for Problem 8.1 using GreedMax can be viewed as an initial pass on the entire training data to distribute subsets of it to compute nodes before the parallel computation begins. Although the proposed algorithm GreedMax is scalable even to very large data sets, it should be noted that there are still computational costs associated with this initialization procedure. However, many statistical learning methods (e.g., the EM algorithm for training Gaussian Mixtures Models,



(a) (b)

Figure 8.1: Comparison between submodular and random partitions for ADMM on 20Newsgroup with $m = 5$ (a) and $m = 10$ (b). For the box plots, the central mark is the median, the box edges are 25th and 75th percentiles, and the bars indicate the best and worst cases.

or the back-propagation algorithm for training deep neural networks) typically involve many training passes over the entire data set. Moreover, during model development (when a researcher is uncertain about the optimal model topology), a number of model topologies need to be tested until the best setting is discovered, even for tools such as [Snoek et al., 2012]. We contend that any cost associated with the initial pass over the data (solving Problem 8.1 for an intelligent partition) can be amortized over the life of the utility of the resulting partition since a smart partition will speed up training time and may even lead to a better overall model in the non-convex case.

8.3 Experimental Results

In this section we empirically evaluate the proposed framework on real-world data partitioning applications including distributed ADMM and distributed deep neural network training.

8.3.1 Distributed Convex Optimization

We first consider data partitioning for distributed convex optimization. We evaluate on a text categorization task. We use 20 Newsgroup data set [1], which consists of 18,774 articles divided almost evenly across 20 classes. The text categorization task is to classify an article into one newsgroup (of twenty) to which it was posted. We randomly split 2/3 and 1/3 of the whole data as the training and test data. The task is solved as a multi-class classification problem, which we formulate as an L-2 regularized logistic regression. We solve this convex optimization problem in a distributive fashion, where the data samples are partitioned and distributed across multiple machines. In particular we implement an ADMM algorithm as described in [Boyd et al., 2011] to solve the distributed convex optimization problem.

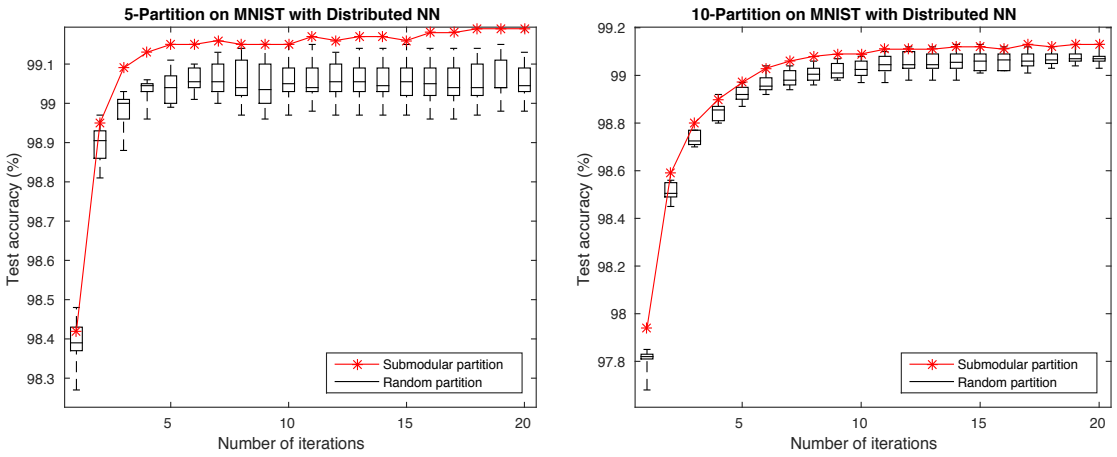
We formulate the data partitioning problem as an robust instance ($\lambda = 0$) of Problem 8.1 under the homogeneous setting. In the experiment, we solve the data partitioning using GreedMax. We model the utility of a data subset using the Naïve Bayes submodular function as defined in Eqn 4.15, which has the form:

$$f_{\text{NB-text}}(S) = \sum_{w \in \mathcal{W}} \sum_{y \in \mathcal{Y}} m_{w,y}(V) \log m_{w,y}(S), \tag{8.3}$$

where $\mathcal{W}$ is the set of all possible words in the documents, $m_{w,y}(A) = \sum_{a \in A} m_{w,y}(a)$ with $m_{w,y}(a)$ counts the number of occurrences of $w \in \mathcal{W}$ in the subset A of documents that are labeled as y. $f_{\text{NB-text}}$ is in the form of a sum of concave over modular functions, hence is monotone submodular [Stobbe and Krause, 2010]. $f_{\text{NB-text}}$ has been shown in Chapter 4 to model the log-likelihood of a data subset for a Naïve Bayes classifier.

We compare the submodular partitioning with the random partitioning for $m = 5$ and $m = 10$. We test with 10 instances of random partitioning. The results are plotted in Fig 8.1. For $m = 10$ we also run an instance on an adversarial partitioning, where each block is formed by grouping every two of the 20 classes in the training data. We observe submodular partitioning converges faster than the random partitioning, both of which perform

[1] Data set is obtained at http://qwone.com/~jason/20Newsgroups/



(a) (b)

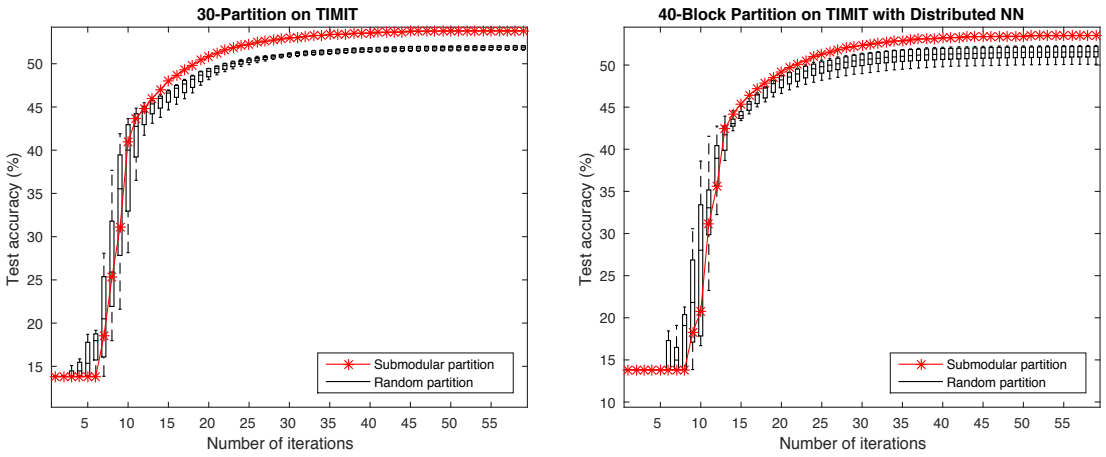
Figure 8.2: Comparison between submodular and random partitions for distributed deep neural nets on MNIST The adversarial partitions are so bad that they are off the plots.

significantly better than the adversarial partition. In particular significant and consistent improvement over the best of 10 random instances is achieved by the submodular partition across all iterations when $m = 5$.

8.3.2 Distributed Deep Neural Networks (DNN)

Next we evaluate our framework on the distributed deep neural network (DNN) training. We test on two tasks: 1) handwritten digit recognition on the MNIST database [2]; 2) phone classification on the TIMIT data. The data for the handwritten digit recognition task consists of 60,000 training and 10,000 test samples. Each data sample is an image of handwritten digit. The training and test data are almost evenly divided into 10 different classes. For the phone classification task, the data consists of 1,124,823 training and 112,487 test samples. Each sample is a frame of speech. The training data is divided into 50 classes, each of which corresponds to a phoneme. The goal of this task to classify each speech sample into one of

[2]Data set is obtained at yann.lecun.com/exdb/mnist



(a) (b)

Figure 8.3: Comparison between submodular and random partitions for distributed deep neural nets on TIMIT. The adversarial partitions are so bad that they are off the plots.

the 50 phone classes.

A 4-layer DNN model is applied for the MNIST experiments, and we train a 5-layered NN for the TIMIT experiments. We apply the same distributed training procedure for both tasks. Given a partitioning of the training data, we distributively solve m instances of sub-problems in each iteration. We define each sub-problem on a separate block of the data. We employ the stochastic gradient descent as the solver on each instance of the sub-problem. In the first iteration we use a randomly generated model as the initial model shared among the m sub-problems. Each sub-problem is solved with 10 epochs of the stochastic gradient decent training. We then average the weights in the m resultant models to obtain a consensus model, which is used as the initial model for each sub-problem in the successive iteration. Note that this distributed training scheme is similar to the one presented in [Povey et al., 2014].

The submodular partitioning for both tasks is obtained by solving the homogeneous case of Problem 8.1 with $\lambda = 0$ using GreedMax on the Nearest Neighbor submodular function

f_{NN} as proposed in Chapter 4. The function is defined as follows:

$$f_{\text{NN}}(A) = \sum_{y \in \mathcal{Y}} \sum_{v \in V^y} \max_{a \in A \cap V^y} s_{v,a}, \tag{8.4}$$

where $s_{v,a}$ is the similarity measure between sample v and a, $\mathcal{Y}$ is the set of class labels, and V^y is the set of samples in V with label $y \in \mathcal{Y}$. Note $\{V^y\}_{y \in \mathcal{Y}}$ forms a disjoint partitioning of the ground set V. In both the MNIST and TIMIT experiments we compute the similarity $s_{v,a}$ as the RBF kernel between the feature representation of v and a. As we have shown in Chapter 4, $f_{\text{c-fac}}$ models the log-likelihood of a data subset for a Nearest Neighbor classifier. In the same chapter, we have also empirically demonstrated the efficacy of f_{NN} in the case of neural network based classifiers.

We run 10 instances of random partitioning as the baseline. As shown in Figure 8.2 and 8.3, the submodular partitioning significantly outperforms the random baseline. For all cases, we observe that better resulting models are obtained by using the submodular partitioning than all 10 instances of the random partitioning. On the other hand, the adversarial partitioning, which is formed by grouping items with the same class, cannot even be trained in both cases.

8.4 Discussion

In this chapter we present a framework for splitting training data intelligently as an initial step to existing parallel statistical learning paradigms. The framework is formulated as a submodular partitioning problem, where we utilize appropriate submodular functions to model the utility of data subsets for training machine learning classifiers. We use a simple and efficient greedy algorithm for solving the submodular partitioning problem. We give empirical validation of the proposed approach on both distributed convex optimization and parallel neural network training across a number of machine learning tasks, including text categorization, handwritten digit recognition, and phone classification. Consistent and significant improvements over the principle of random partitioning are achieved by the proposed intelligent data distribution framework.

Although the empirical success on the TIMIT data set that has over a million training samples was shown here, this is still a proof of concept experiment as the TIMIT data is still able to reside on a single compute. Future work will concentrate on evaluating the framework on even larger scale of problems for which the parallel training scheme is more suitable. Another direction for the future work is to consider more explicit constraint for modeling the balance of the computational costs associated with each block in a resultant partitioning.

Chapter 9

CONCLUSIONS

This thesis has presented a data summarization and a data partitioning paradigm for addressing the big data challenge using the tool of submodular function optimization. For both paradigms, we concentrate on two components: modeling and optimization. We argue that submodularity naturally addresses both components. In our case studies for data summarization and data partitioning (Chapter 2, 3, 4 and 8) we demonstrate that submodular functions naturally occur as the model for the notion of utility and information for these real-world tasks and moreover, our proposed framework yields significantly better performance over the existing baseline approaches across these tasks. To further improve the goodness of a submodular function as the utility model, we give a novel "interactive" learning algorithmic framework, which enables learning the mixture of submodular functions more feasible and scalable (Chapter 5). Our algorithmic contributions for scaling up the submodular data summarization and data partitioning paradigms are presented in Chapter 6 and 7, where we proposed several new algorithms that not only scale to large data sets but also achieve theoretical guarantees comparable to the state-of-the-art. We conclude with several directions of research as the future work of this thesis.

- One interesting direction of research is to show further connection of submodular optimization to the increasingly popular area of deep neural networks (DNN). In Chapter 4 and 8 we have leveraged the use of submodular optimization to several DNN related tasks. We demonstrated the success of the submodular data summarization paradigm for the task of batch active learning of DNN models (Chapter 4) as well as the effectiveness of the submodular data partitioning paradigm on the task of parallel training of DNN's (Chapter 8). It would be interesting, in the future, to investigate many other

connections between these two technologies.

One simple idea would be to utilize deep learning models to featurize data which is then applied to instantiate the submodular functions. One may significantly improve the empirical performance of the submodular function based paradigms on the case studies considered in this thesis by instantiating the submodular utility functions via the DNN based features.

Another idea is to apply submodular function based data processing techniques to arrange the data for better training of DNN models, in addition to using the submodular data partitioning method for improving DNN parallel training problems. One idea would be to devise a scheme to manipulate the order of the training data so that faster training of DNN models may be achieved on both the single compute and the parallel computing settings.

It would also be of interest to apply submodular optimization to optimize DNN structures. Motivated by [Mariet and Sra, 2015] where they show empirical success of using Determinantal Point Processes (DPP) to diversify a learned model by pruning out redundant neurons , it would be worthwhile investigating whether similar or even better performance improvement of this idea may be achieved by using the submodular data summarization paradigm proposed in this thesis.

- In this thesis, we primarily focus on submodular optimization problems and their applications. The optimization problems are, in general, in terms of discrete set functions. This optimization setting can be significantly generalized if the optimization objective involves both discrete and continuous variables. To make it more concrete, define a function $f : 2^V \times \mathbb{R}^d \rightarrow \mathbb{R}$ that maps a set $A \subseteq V$ and a real-valued vector $x \in \mathbb{R}^d$ to a real value $f(A, x)$, the generalized optimization problem can be stated as follows:

$$\max_{A \in \mathcal{C}, x \in \mathcal{D}} f(A, x), \tag{9.1}$$

where $\mathcal{C} \subseteq 2^V$ defines the set of all feasible sets for A, and $\mathcal{D} \subseteq \mathbb{R}^d$ defines the feasible region of the continuous variable x. Problem 9.1, in the most general form, is extremely hard to optimize with any performance guarantee. The open question would be: "Are there approximation algorithms for Problem 9.1 when f is submodular in A and concave in x, and the feasibility constraints $\mathcal{C}$ and $\mathcal{D}$ can be defined via a matroid and a convex set, respectively?" Moreover, it would be of great interest to examine whether the applications considered in this thesis could be formulated in terms of this general optimization problem.

BIBLIOGRAPHY

[Agarwal et al., 2005] Agarwal, P. K., Har-Peled, S., and Varadarajan, K. R. (2005). Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30.

[Alon et al., 2011] Alon, N., Arora, S., Manokaran, R., Moshkovitz, D., and Weinstein, O. (2011). Inapproximability of densest k-subgraph from average case hardness. *Manuscript, available at www. cs. princeton. edu/˜ rajsekar/papers/dks. pdf*.

[Andrés et al., 1999] Andrés, M. E., Burger, C., Peral-Rubio, M. J., Battaglioli, E., Anderson, M. E., Grimes, J., Dallman, J., Ballas, N., and Mandel, G. (1999). Corest: a functional corepressor required for regulation of neural-specific gene expression. *Proceedings of the National Academy of Sciences*, 96(17):9873–9878.

[Asadpour and Saberi, 2010] Asadpour, A. and Saberi, A. (2010). An approximation algorithm for max-min fair allocation of indivisible goods. In *SICOMP*.

[Bach, 2011] Bach, F. (2011). Learning with submodular functions: A convex optimization perspective. *arXiv preprint arXiv:1111.6453*.

[Badanidiyuru et al., 2014] Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., and Krause, A. (2014). Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 671–680. ACM.

[Badanidiyuru and Vondrák, 2014] Badanidiyuru, A. and Vondrák, J. (2014). Fast algorithms for maximizing submodular functions. In *SODA*.

[Bagon, 2006] Bagon, S. (2006). Matlab wrapper for graph cut.

[Bai et al., 2016] Bai, W., Iyer, R., Wei, K., and Bilmes, J. (2016). Algorithms for optimizing the ratio of submodular functions. In *International Conference on Machine Learning (ICML)*, New York, USA.

[Bairi et al., 2015] Bairi, R., Iyer, R. K., Ramakrishnan, G., and Bilmes, J. A. (2015). Summarization of multi-document topic hierarchies using submodular mixtures. In *ACL*.

[Balagani and Phoha, 2010] Balagani, K. S. and Phoha, V. V. (2010). On the feature selection criterion based on an approximation of multidimensional mutual information. *PAMI, IEEE Transactions*.

[Balcan and Harvey, 2011] Balcan, N. and Harvey, N. (2011). Learning submodular functions. In *STOC*.

[Barbosa et al., 2015] Barbosa, R., Ene, A., Nguyen, H. L., and Ward, J. (2015). The power of randomization: Distributed submodular maximization on massive datasets. *arXiv preprint arXiv:1502.02606*.

[Bateni et al., 2013] Bateni, M., Hajiaghayi, M., and Zadimoghaddam, M. (2013). Submodular secretary problem and extensions. *ACM Transactions on Algorithms (TALG)*, 9(4):32.

[Bernstein et al., 2010] Bernstein, B. E., Stamatoyannopoulos, J. A., Costello, J. F., Ren, B., Milosavljevic, A., Meissner, A., Kellis, M., Marra, M. A., Beaudet, A. L., Ecker, J. R., Farnham, P. J., Hirst, M., Lander, E. S., Mikkelsen, T. S., and Thomson, J. A. (2010). The NIH Roadmap Epigenomics Mapping Consortium. *Nature Biotechnology*, 28(10):1045–1048.

[Beygelzimer et al., 2006] Beygelzimer, A., Kakade, S., and Langford, J. (2006). Cover trees for nearest neighbor. In *ICML*.

[Bhatia et al., 2010] Bhatia, N. et al. (2010). Survey of nearest neighbor techniques. *arXiv preprint arXiv:1007.0085*.

[Boiman et al., 2008] Boiman, O., Shechtman, E., and Irani, M. (2008). In defense of nearest-neighbor based image classification. In *CVPR*, pages 1–8. IEEE.

[Boyd et al., 2011] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge Univ Pr.

[Boykov and Jolly, 2001] Boykov, Y. and Jolly, M. (2001). Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *ICCV*.

[Boykov and Kolmogorov, 2004] Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137.

[Buchbinder and Feldman, 2015] Buchbinder, N. and Feldman, M. (2015). Deterministic algorithms for submodular maximization problems. *arXiv preprint arXiv:1508.02157*.

[Buchbinder et al., 2012] Buchbinder, N., Feldman, M., Naor, J., and Schwartz, R. (2012). A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *FOCS*.

[Buchbinder et al., 2014] Buchbinder, N., Feldman, M., Naor, J. S., and Schwartz, R. (2014). Submodular maximization with cardinality constraints. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1433–1452. SIAM.

[Buchbinder et al., 2015] Buchbinder, N., Feldman, M., and Schwartz, R. (2015). Online submodular maximization with preemption. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1202–1216. SIAM.

[Burgess-Beusse et al., 2002] Burgess-Beusse, B., Farrell, C., Gaszner, M., Litt, M., Mutskov, V., Recillas-Targa, F., Simpson, M., West, A., and Felsenfeld, G. (2002). The insulation of genes from external enhancers and silencing chromatin. *Proceedings of the National Academy of Sciences of the United States of America*, 99(Suppl 4):16433.

[Byrnes, 2015] Byrnes, K. M. (2015). A tight analysis of the submodular–supermodular procedure. *Discrete Applied Mathematics*, 186:275–282.

[Calinescu et al., 2007] Calinescu, G., Chekuri, C., Pal, M., and Vondrák., J. (2007). Maximizing a monotone submodular function under a matroid constraint. *IPCO*.

[Chakrabarty et al., 2014] Chakrabarty, D., Jain, P., and Kothari, P. (2014). Provable submodular minimization using wolfe's algorithm. In *Advances in Neural Information Processing Systems*, pages 802–809.

[Chaudhuri et al., 2015] Chaudhuri, K., Kakade, S. M., Netrapalli, P., and Sanghavi, S. (2015). Convergence rates of active learning for maximum likelihood estimation. In *NIPS*.

[Chekuri and Ene, 2011a] Chekuri, C. and Ene, A. (2011a). Approximation algorithms for submodular multiway partition. In *FOCS*.

[Chekuri and Ene, 2011b] Chekuri, C. and Ene, A. (2011b). Submodular cost allocation problem and applications. In *Automata, Languages and Programming*, pages 354–366. Springer.

[Chen and Krause, 2013] Chen, Y. and Krause, A. (2013). Near-optimal batch mode active learning and adaptive submodular optimization. In *ICML*, pages 160–168.

[Chierichetti et al., 2010] Chierichetti, F., Kumar, R., and Tomkins, A. (2010). Max-cover in Map-Reduce. In *WWW*.

[Conforti and Cornuejols, 1984] Conforti, M. and Cornuejols, G. (1984). Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Applied Mathematics*.

[Consortium, 2014] Consortium, U. (2014). UniProt: a hub for protein information. *Nucleic acids research*, page gku989.

[Craven and Bockhorst, 2005] Craven, M. and Bockhorst, J. (2005). Markov networks for detecting overlapping elements in sequence data. *Advances in Neural Information Processing Systems*, 17:193.

[Cunningham, 1984] Cunningham, W. H. (1984). Testing membership in matroid polyhedra. *Journal of Combinatorial Theory, Series B*, 36(2):161–188.

[Cuong et al., 2010] Cuong, N. V., Lee, W. S., and Ye, N. (2010). Near-optimal adaptive pool-based active learning with general loss. *UAI*.

[Das and Kempe, 2011] Das, A. and Kempe, D. (2011). Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv preprint arXiv:1102.3975*.

[Davis and Goadrich, 2006] Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *Proceedings of the International Conference on Machine Learning*.

[Day et al., 2007] Day, N., Hemmaplardh, A., Thurman, R. E., Stamatoyannopoulos, J. A., and Noble, W. S. (2007). Unsupervised segmentation of continuous genomic data. *Bioinformatics*, 23(11):1424–1426.

[Dobzinski and Mor, 2015] Dobzinski, S. and Mor, A. (2015). A deterministic algorithm for maximizing submodular functions. *arXiv preprint arXiv:1507.07237*.

[Edmonds, 1970] Edmonds, J. (1970). Submodular functions, matroids and certain polyhedra. *Combinatorial structures and their Applications*.

[El-Arini et al., 2009] El-Arini, K., Veda, G., Shahaf, D., and Guestrin, C. (2009). Turning down the noise in the blogosphere. In *SIGKDD*.

[ENCODE Project Consortium, 2012] ENCODE Project Consortium (2012). An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489:57–74.

[Ene et al., 2013] Ene, A., Vondrák, J., and Wu, Y. (2013). Local distribution and the symmetry gap: Approximability of multiway partitioning problems. In *SODA*.

[Ernst and Kellis, 2010] Ernst, J. and Kellis, M. (2010). Discovery and characterization of chromatin states for systematic annotation of the human genome. *Nature Biotechnology*, 28(8):817–825.

[Fan et al., 2008] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874.

[Feige, 1998] Feige, U. (1998). A threshold of ln n for approximating set cover. *JACM*.

[Feige, 2002] Feige, U. (2002). Relations between average case complexity and approximation complexity. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 534–543. ACM.

[Feige et al., 2011] Feige, U., Mirrokni, V., and Vondrák, J. (2011). Maximizing non-monotone submodular functions. *SIAM J. COMPUT.*, 40(4):1133–1155.

[Feige et al., 1997] Feige, U., Seltser, M., et al. (1997). *On the densest k-subgraph problem*. Citeseer.

[Feldman et al., 2011] Feldman, M., Naor, J., and Schwartz, R. (2011). A unified continuous greedy algorithm for submodular maximization. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 570–579. IEEE.

[Filion et al., 2010] Filion, G. J., van Bemmel, J. G., Braunschweig, U., Talhout, W., Kind, J., Ward, L. D., Brugman, W., de Castro, I. J., Kerkhoven, R. M., Bussemaker, H. J., and van Steensel, B. (2010). Systematic protein location mapping reveals five principal chromatin types in *Drosophila* cells. *Cell*, 143(2):212–224.

[Fisher et al., 1978] Fisher, M., Nemhauser, G., and Wolsey, L. (1978). An analysis of approximations for maximizing submodular set functions—II. *Polyhedral combinatorics*, pages 73–87.

[Fujishige, 2005] Fujishige, S. (2005). *Submodular functions and optimization*. Elsevier Science Ltd.

[Fujishige et al., 2006] Fujishige, S., Hayashi, T., and Isotani, S. (2006). *The minimum-norm-point algorithm applied to submodular function minimization and linear programming*. Citeseer.

[Fujishige and Isotani, 2011] Fujishige, S. and Isotani, S. (2011). A submodular function minimization algorithm based on the minimum-norm base. *Pacific Journal of Optimization*, 7:3–17.

[Gabillon et al., 2013] Gabillon, V., Kveton, B., Wen, Z., Eriksson, B., and Muthukrishnan, S. (2013). Adaptive submodular maximization in bandit setting. In *NIPS*.

[Gharan and Vondrák, 2011] Gharan, S. O. and Vondrák, J. (2011). Submodular maximization by simulated annealing. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1098–1116. SIAM.

[Goel et al., 2009] Goel, G., Tripathi, P., and Wang, L. (2009). Optimal approximation algorithms for multi-agent combinatorial problems with discounted price functions. *arXiv preprint arXiv:0911.1346*.

[Goemans et al., 2009] Goemans, M., Harvey, N., Iwata, S., and Mirrokni, V. (2009). Approximating submodular functions everywhere. In *SODA*, pages 535–544.

[Goemans and Williamson, 1995] Goemans, M. X. and Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145.

[Golovin, 2005] Golovin, D. (2005). Max-min fair allocation of indivisible goods. *Technical Report CMU-CS-05-144*.

[Golovin and Krause, 2010] Golovin, D. and Krause, A. (2010). Adaptive submodularity: A new approach to active learning and stochastic optimization. In *COLT*.

[Gomes and Krause, 2010] Gomes, R. and Krause, A. (2010). Budgeted nonparametric learning from data streams. In *ICML*.

[Grötschel et al., 1981] Grötschel, M., Lovász, L., and Schrijver, A. (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197.

[Gu et al., 2012] Gu, Q., Zhang, T., Han, J., and Ding, C. H. (2012). Selective labeling via error bound minimization. In *NIPS*.

[Guillory and Bilmes, 2010] Guillory, A. and Bilmes, J. (2010). Interactive submodular set cover. *ICML*.

[Guillory and Bilmes, 2011] Guillory, A. and Bilmes, J. (2011). Active semi-supervised learning using submodular functions. In *UAI*.

[Gupta et al., 2010] Gupta, A., Roth, A., Schoenebeck, G., and Talwar, K. (2010). Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *Internet and Network Economics*, pages 246–257. Springer.

[Gygli et al., 2015] Gygli, M., Grabner, H., and Van Gool, L. (2015). Video summarization by learning submodular mixtures of objectives. In *CVPR*.

[Hakkani-Tur et al., 2002] Hakkani-Tur, D., Riccardi, G., and Gorin, A. (2002). Active learning for automatic speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, pages IV–3904.

[Hochbaum and Shmoys, 1988] Hochbaum, D. S. and Shmoys, D. B. (1988). A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. In *SICOMP*.

[Hoffman et al., 2012] Hoffman, M. M., Buske, O. J., Wang, J., Weng, Z., Bilmes, J. A., and Noble, W. S. (2012). Unsupervised pattern discovery in human chromatin structure through genomic segmentation. *Nature Methods*, 9(5):473–476.

[Hoffman et al., 2013] Hoffman, M. M., Ernst, J., Wilder, S. P., Kundaje, A., Harris, R. S., Libbrecht, M., Giardine, B., Ellenbogen, P. M., Bilmes, J. A., Birney, E., Hardison, R. C., Dunham, I., Kellis, M., and Noble, W. S. (2013). Integrative annotation of chromatin elements from ENCODE data. *Nucleic Acids Res*, 41(2):827–41.

[Hoi et al., 2006] Hoi, S. C., Jin, R., Zhu, J., and Lyu, M. R. (2006). Batch mode active learning and its application to medical image classification. In *ICML*.

[Huang et al., 2010] Huang, S.-J., Jin, R., and Zhou, Z.-H. (2010). Active learning by querying informative and representative examples. In *NIPS*.

[Itoh et al., 2012] Itoh, N., Sainath, T. N., Jiang, D. N., Zhou, J., and Ramabhadran, B. (2012). N-best entropy based data selection for acoustic modeling. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4133–4136.

[Iwata et al., 2001] Iwata, S., Fleischer, L., and Fujishige, S. (2001). A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777.

[Iwata and Nagano, 2009] Iwata, S. and Nagano, K. (2009). Submodular function minimization under covering constraints. In *In FOCS*, pages 671–680. IEEE.

[Iyer and Bilmes, 2012] Iyer, R. and Bilmes, J. (2012). Algorithms for approximate minimization of the difference between submodular functions, with applications. In *Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, USA. AUAI.

[Iyer and Bilmes, 2013] Iyer, R. and Bilmes, J. (2013). Submodular optimization with submodular cover and submodular knapsack constraints. In *NIPS*.

[Iyer and Bilmes, 2015] Iyer, R. and Bilmes, J. (2015). Polyhedral aspects of submodularity, convexity and concavity. *arXiv preprint arXiv:1506.07329*.

[Iyer et al., 2013a] Iyer, R., Jegelka, S., and Bilmes, J. (2013a). Curvature and Optimal Algorithms for Learning and Minimizing Submodular Functions. *NIPS*.

[Iyer et al., 2013b] Iyer, R., Jegelka, S., and Bilmes, J. (2013b). Fast semidifferential based submodular function optimization. In *ICML*.

[Iyer et al., 2014] Iyer, R., Jegelka, S., and Bilmes, J. (2014). Monotone closure of relaxed constraints in submodular optimization: Connections between minimization and maximization.

[Iyer, 2015] Iyer, R. K. (2015). *Submodular Optimization and Machine Learning: Theoretical Results, Unifying and Scalable Algorithms, and Applications*. PhD thesis.

[Jegelka and Bilmes, 2011] Jegelka, S. and Bilmes, J. A. (2011). Submodularity beyond submodular energies: coupling edges in graph cuts. In *CVPR*.

[Jia et al., 2014] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

[Johnson, 1949] Johnson, N. L. (1949). Systems of frequency curves generated by methods of translation. *Biometrika*, pages 149–176.

[Kawahara et al., 2011] Kawahara, Y., Nagano, K., and Okamoto, Y. (2011). Submodular fractional programming for balanced clustering. *Pattern Recognition Letters*, 32(2):235–243.

[Kemp and Waibel, 1998] Kemp, T. and Waibel, A. (1998). Unsupervised training of a speech recognizer using TV broadcasts. In *ICSLP*, volume 98, pages 2207–2210.

[Kempe et al., 2003] Kempe, D., Kleinberg, J., and Tardos, É. (2003). Maximizing the spread of influence through a social network. In *Proc. SIGKDD*, pages 137–146. ACM.

[Khot and Ponnuswami, 2007] Khot, S. and Ponnuswami, A. (2007). Approximation algorithms for the max-min allocation problem. In *APPROX*.

[Kohli et al., 2013] Kohli, P., Osokin, A., and Jegelka, S. (2013). A principled deep random field model for image segmentation. In *CVPR*.

[Kolmogorov and Zabih, 2004] Kolmogorov, V. and Zabih, R. (2004). What energy functions can be minimized via graph cuts? *IEEE TPAMI*, 26(2):147–159.

[Krause and Guestrin, 2005a] Krause, A. and Guestrin, C. (2005a). Near-optimal nonmyopic value of information in graphical models. In *UAI*.

[Krause and Guestrin, 2005b] Krause, A. and Guestrin, C. (2005b). A note on the budgeted maximization of submodular functions.

[Krause et al., 2008a] Krause, A., McMahan, B., Guestrin, C., and Gupta, A. (2008a). Robust submodular observation selection. *Journal of Machine Learning Research (JMLR)*, 9:2761–2801.

[Krause et al., 2008b] Krause, A., Singh, A., and Guestrin, C. (2008b). Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, 9:235–284.

[Kulesza and Taskar, 2012] Kulesza, A. and Taskar, B. (2012). Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2-3):123–286.

[Kulik et al., 2009] Kulik, A., Shachnai, H., and Tamir, T. (2009). Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 545–554. Society for Industrial and Applied Mathematics.

[Kumar and Andreou, 1997] Kumar, N. and Andreou, A. G. (1997). *Investigation of silicon auditory models and generalization of linear discriminant analysis for improved speech recognition*. PhD thesis, Johns Hopkins University.

[Kumar et al., 2013] Kumar, R., Moseley, B., Vassilvitskii, S., and Vattani, A. (2013). Fast greedy algorithms in mapreduce and streaming. In *SPAA*.

[Kundaje et al., 2015] Kundaje, A., Meuleman, W., Ernst, J., Bilenky, M., Yen, A., Heravi-Moussavi, A., Kheradpour, P., Zhang, Z., Wang, J., Ziller, M. J., et al. (2015). Integrative analysis of 111 reference human epigenomes. *Nature*, 518(7539):317–330.

[Lamel et al., 2002] Lamel, L., Gauvain, J.-L., and Adda, G. (2002). Lightly supervised and unsupervised acoustic model training. *Computer Speech & Language*, 16(1):115–129.

[Landt et al., 2012] Landt, S. G., Marinov, G. K., Kundaje, A., Kheradpour, P., Pauli, F., Batzoglou, S., Bernstein, B. E., Bickel, P., Brown, J. B., Cayting, P., Chen, Y., Desalvo, G., Epstein, C., Fisher-Aylor, K. I., Euskirchen, G., Gerstein, M., Gertz, J., Hartemink, A. J., Hoffman, M. M., Iyer, V. R., Jung, Y. L., Karmakar, S., Kellis, M., Kharchenko, P. V., Li, Q., Liu, T., Liu, X. S., Ma, L., Milosavljevic, A., Myers, R. M., Park, P. J., Pazin, M. J., Perry, M. D., Raha, D., Reddy, T. E., Rozowsky, J., Shoresh, N., Sidow, A., Slattery, M., Stamatoyannopoulos, J. A., Tolstorukov, M. Y., White, K. P., Xi, S., Farnham, P. J., Lieb, J. D., Wold, B. J., and Snyder, M. (2012). ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. *Genome Research*, 22(9):1813–1831.

[Lang, 1995] Lang, K. (1995). Newsweeder: Learning to filter netnews. In *ICML*.

[Lee et al., 2009] Lee, J., Mirrokni, V., Nagarajan, V., and Sviridenko, M. (2009). Non-monotone submodular maximization under matroid and knapsack constraints. In *STOC*, pages 323–332. ACM.

[Lee et al., 2010] Lee, J., Mirrokni, V. S., Nagarajan, V., and Sviridenko, M. (2010). Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM Journal on Discrete Mathematics*, 23(4):2053–2078.

[Lenstra et al., 1990] Lenstra, J. K., Shmoys, D. B., and Tardos, É. (1990). Approximation algorithms for scheduling unrelated parallel machines. In *Mathematical programming*.

[Leskovec et al., 2007] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. (2007). Cost-effective outbreak detection in networks. In *SIGKDD*.

[Lewis and Gale, 1994] Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proc. SIGIR*, pages 3–12. Springer-Verlag New York, Inc.

[Li et al., 2007] Li, B., Carey, M., and Workman, J. L. W. (2007). The role of chromatin during transcription. *Cell*, 128(4):707–719.

[Li et al., 2010] Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *WWW*.

[Li et al., 2015] Li, M., Andersen, D., and Smola, A. (2015). Graph partitioning via parallel submodular approximation to accelerate distributed machine learning. In *arXiv preprint arXiv:1505.04636*.

[Lian et al., 2008] Lian, H., Thompson, W., Thurman, R. E., Stamatoyannopoulos, J. A., Noble, W. S., and Lawrence, C. (2008). Automated mapping of large-scale chromatin structure in ENCODE. *Bioinformatics*, 24(17):1911–1916.

[Libbrecht et al., 2015] Libbrecht, M., Ay, F., Hoffman, M. M., Gilbert, D. M., Bilmes, J. A., and Noble, W. S. (2015). Joint annotation of chromatin state and chromatin conformation reveals relationships among domain types and identifies domains of cell-type-specific expression. *Genome Research*, 25(4):544–557.

[Lin and Bilmes, 2011] Lin, H. and Bilmes, J. (2011). 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.

[Lin and Bilmes, 2012] Lin, H. and Bilmes, J. (2012). Learning mixtures of submodular shells with application to document summarization. In *Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, USA. AUAI.

[Lin et al., 2009] Lin, H., Bilmes, J., and Xie, S. (2009). Graph-based submodular selection for extractive summarization. In *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, pages 381–386. IEEE.

[Lin and Bilmes, 2009] Lin, H. and Bilmes, J. A. (2009). How to select a good training-data subset for transcription: Submodular active selection for sequences. In *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Brighton, UK.

[Lindgren et al.,] Lindgren, E. M., Wu, S., and Dimakis, A. G. Sparse and greedy: Sparsifying submodular facility location problems.

[Liu et al., 2007] Liu, S.-H., Chu, F.-H., Lin, S.-H., Lee, H.-S., and Chen, B. (2007). Training data selection for improving discriminative training of acoustic models. In *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 284–289. IEEE.

[Liu et al., 2013] Liu, Y., Wei, K., Kirchhoff, K., Song, Y., and Bilmes, J. (2013). Submodular feature selection for high-dimensional acoustic score spaces. In *ICASSP*.

[Lovász, 1983] Lovász, L. (1983). Submodular functions and convexity. *Mathematical Programming*.

[MacKay, 1992] MacKay, D. J. (1992). Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604.

[Mariet and Sra, 2015] Mariet, Z. and Sra, S. (2015). Diversity networks. *arXiv preprint arXiv:1511.05077*.

[Mei et al., 2016] Mei, J., Zhang, H., and Lu, B.-L. (2016). On the reducibility of submodular functions. *arXiv preprint arXiv:1601.00393*.

[Minoux, 1978] Minoux, M. (1978). Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, pages 234–243.

[Mirchandani and Francis, 1990] Mirchandani, P. B. and Francis, R. L. (1990). *Discrete Location Theory*. Wiley.

[Mirzasoleiman et al., 2015] Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrak, J., and Krause, A. (2015). Lazier than lazy greedy. In *Proc. AAAI*.

[Mirzasoleiman et al., 2013] Mirzasoleiman, B., Karbasi, A., Sarkar, R., and Krause, A. (2013). Distributed submodular maximization: Identifying representative elements in massive data. In *NIPS*.

[Nagano et al., 2011] Nagano, K., Kawahara, Y., and Aihara, K. (2011). Size-constrained submodular minimization through minimum norm base. In *ICML*.

[Narasimhan and Bilmes, 2005] Narasimhan, M. and Bilmes, J. (2005). A submodular-supermodular procedure with applications to discriminative structure learning. In *UAI*.

[Narasimhan et al., 2005] Narasimhan, M., Jojic, N., and Bilmes, J. A. (2005). Q-clustering. In *NIPS*.

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

[Nemhauser et al., 1978] Nemhauser, G., Wolsey, L., and Fisher, M. (1978). An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294.

[Nguyen and Smeulders, 2004] Nguyen, H. T. and Smeulders, A. (2004). Active learning using pre-clustering. In *ICML*, page 79. ACM.

[Nigam et al., 2000] Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134.

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

[Pan et al., 2014] Pan, X., Jegelka, S., Gonzalez, J. E., Bradley, J. K., and Jordan, M. I. (2014). Parallel double greedy submodular maximization. In *Advances in Neural Information Processing Systems*, pages 118–126.

[Peng et al., 2005] Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *PAMI, IEEE Transactions*.

[Povey et al., 2011] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. (2011). The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*.

[Povey et al., 2005] Povey, D., Kingsbury, B., Mangu, L., Saon, G., Soltau, H., and Zweig, G. (2005). fMPE: Discriminatively trained features for speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 961–964.

[Povey and Woodland, 2002] Povey, D. and Woodland, P. C. (2002). Minimum phone error and i-smoothing for improved discriminative training. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages I–105.

[Povey et al., 2014] Povey, D., Zhang, X., and Khudanpur, S. (2014). Parallel training of deep neural networks with natural gradient and parameter averaging. *arXiv preprint arXiv:1410.7455*.

[Prasad et al., 2014] Prasad, A., Jegelka, S., and Batra, D. (2014). Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. In *NIPS*.

[Qin et al., 2014] Qin, L., Chen, S., and Zhu, X. (2014). Contextual combinatorial bandit and its application on diversified online recommendation. In *SDM*.

[Queyranne, 1998] Queyranne, M. (1998). Minimizing symmetric submodular functions. *Mathematical Programming*, 82(1-2):3–12.

[Radlinski et al., 2008] Radlinski, F., Kleinberg, R., and Joachims, T. (2008). Learning diverse rankings with multi-armed bandits. In *ICML*.

[Raman et al., 2012] Raman, K., Shivaswamy, P., and Joachims, T. (2012). Online learning to diversify from implicit feedback. In *SIGKDD*.

[Reed and Ghahramani, 2013] Reed, C. and Ghahramani, Z. (2013). Scaling the indian buffet process via submodular maximization. *arXiv preprint arXiv:1304.3285*.

[Rousu and Shawe-Taylor, 2006] Rousu, J. and Shawe-Taylor, J. (2006). Efficient computation of gapped substring kernels on large alphabets. *Journal of Machine Learning Research*, 6(2):1323.

[Schalkwyk et al., 2010] Schalkwyk, J., Beeferman, D., Beaufays, F., Byrne, B., Chelba, C., Cohen, M., Kamvar, M., and Strope, B. (2010). your word is my command: Google search by voice: A case study. In *Advances in Speech Recognition*, pages 61–90. Springer.

[Schrijver, 2000] Schrijver, A. (2000). A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355.

[scienceexchange, 2015] scienceexchange (2015). ChIP-seq prices at scienceexchange.com. https://www.scienceexchange.com/services/chip-seq.

[Settles, 2009] Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

[Settles, 2010] Settles, B. (2010). Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11.

[Seung et al., 1992] Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM.

[Shah et al., 2011] Shah, R., Iyer, R., and Chaudhuri, S. (2011). Object mining for large video data. *Proc. BMVC*, 22(10):761–767.

[Shamaiah et al., 2010] Shamaiah, M., Banerjee, S., and Vikalo, H. (2010). Greedy sensor selection: Leveraging submodularity. In *Proc. CDC*, pages 2572–2577. IEEE.

[Shinohara, 2014] Shinohara, Y. (2014). A submodular optimization approach to sentence set selection. In *ICASSP*, pages 4112–4115. IEEE.

[Shioura, 2009] Shioura, A. (2009). On the pipage rounding algorithm for submodular function maximizationa view from discrete convex analysis. *Discrete Mathematics, Algorithms and Applications*, 1(01):1–23.

[Singer, 2012] Singer, Y. (2012). How to win friends and influence people, truthfully: influence maximization mechanisms for social networks. In *WSDM*. ACM.

[Singla et al., 2014] Singla, A., Bogunovic, I., Bartók, G., Karbasi, A., and Krause, A. (2014). Near-optimally teaching the crowd to classify. *arXiv preprint arXiv:1402.2092*.

[Singla et al., 2016] Singla, A., Tschiatschek, S., and Krause, A. (2016). Noisy Submodular Maximization via Adaptive Sampling with Applications to Crowdsourced Image Collection Summarization. In *AAAI*.

[Siohan, 2014] Siohan, O. (2014). Training data selection based on context-dependent state matching. In *ICASSP*.

[Sipos et al., 2011] Sipos, R., Shivaswamy, P., and Joachims, T. (2011). Large-margin learning of submodular summarization methods. *arXiv preprint arXiv:1110.2162*.

[Snoek et al., 2012] Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

[Stobbe and Krause, 2010] Stobbe, P. and Krause, A. (2010). Efficient minimization of decomposable submodular functions. In *NIPS*.

[Sviridenko, 2004] Sviridenko, M. (2004). A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43.

[Svitkina and Fleischer, 2008] Svitkina, Z. and Fleischer, L. (2008). Submodular approximation: Sampling-based algorithms and lower bounds. In *FOCS*, pages 697–706.

[Svitkina and Fleischer, 2011] Svitkina, Z. and Fleischer, L. (2011). Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM Journal on Computing*, 40(6):1715–1737.

[Thompson and Parthasarathy, 2006] Thompson, S. E. and Parthasarathy, S. (2006). Moore's law: the future of si microelectronics. *materials today*, 9(6):20–25.

[Thurman et al., 2007] Thurman, R. E., Day, N., Noble, W. S., and Stamatoyannopoulos, J. A. (2007). Identification of higher-order functional domains in the human ENCODE regions. *Genome Research*, 17:917–927.

[Tschiatschek et al., 2014] Tschiatschek, S., Iyer, R. K., Wei, H., and Bilmes, J. A. (2014). Learning mixtures of submodular functions for image collection summarization. In *NIPS*.

[Visel et al., 2009] Visel, A., Blow, M. J., Li, Z., Zhang, T., Akiyama, J. A., Holt, A., Plajzer-Frick, I., Shoukry, M., Wright, C., Chen, F., Afzal, V., Ren, B., Rubin, E. M., and Pennacchio, L. A. (2009). ChIP-seq accurately predicts tissue-specific activity of enhancers. *Nature*, 457(7231):854–858.

[Von Luxburg, 2007] Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416.

[Vondrák, 2007] Vondrák, J. (2007). *Submodularity in combinatorial optimization*. PhD thesis, PhD thesis, Charles University, Prague, Czech Republic.

[Vondrák, 2008] Vondrák, J. (2008). Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*.

[Vondrák, 2010] Vondrák, J. (2010). Submodularity and curvature: the optimal algorithm. *RIMS Kokyuroku Bessatsu*, 23.

[Vondrák, 2013] Vondrák, J. (2013). Symmetry and approximability of submodular maximization problems. *SIAM Journal on Computing*, 42(1):265–304.

[Wei et al., 2015] Wei, K., Iyer, R., and Bilmes, J. (2015). Submodularity in data subset selection and active learning. In *ICML*.

[Wei et al., 2016] Wei, K., Libbrecht, M. W., Bilmes, J. A., and Noble, W. (2016). Choosing panels of genomics assays using submodular optimization. *bioRxiv*, page 036137.

[Wei et al., 2014] Wei, K., Liu, Y., Kirchhoff, K., Bartels, C., and Bilmes, J. (2014). Submodular subset selection for large-scale speech training data. In *ICASSP*.

[Wei et al., 2013] Wei, K., Liu, Y., Kirchhoff, K., and Bilmes, J. (2013). Using document summarization techniques for speech data subset selection. In *NAACL/HLT*.

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

[Wu et al., 2007] Wu, Y., Zhang, R., and Rudnicky, A. (2007). Data selection for speech recognition. In *ASRU*.

[Xu et al., 2003] Xu, Z., Yu, K., Tresp, V., Xu, X., and Wang, J. (2003). *Representative sampling for text classification using support vector machines*. Springer.

[Yue and Guestrin, 2011] Yue, Y. and Guestrin, C. (2011). Linear submodular bandits and their application to diversified retrieval. In *NIPS*.

[Zhang et al., 2008] Zhang, Y., Liu, T., Meyer, C. A., Eeckhoute, J., Johnson, D. S., Bernstein, B. E., Nusbaum, C., Myers, R. M., Brown, M., Li, W., and Liu, X. S. (2008). Model-based analysis of ChIP-Seq (MACS). *Genome Biology*, 9(9):R137.

[Zhao et al., 2004] Zhao, L., Nagamochi, H., and Ibaraki, T. (2004). On generalized greedy splitting algorithms for multiway partition problems. *Discrete applied mathematics*, 143(1):130–143.

[Zheng et al., 2014] Zheng, J., Jiang, Z., Chellappa, R., and Phillips, J. P. (2014). Submodular attribute selection for action recognition in video. In *NIPS*.

Appendix A

APPENDIX

A.1 Table of the 10 most frequently performed assay types

	Assay Type
1	DnaseSeq
2	H3K4me3
3	CTCF
4	FaireSeq
5	POLR2A
6	H3K27me3
7	H3K36me3
8	H3K79me2
9	H3K4me2
10	H3K9me3

Table A.1: The 10 most frequently performed assay types.

A.2 Derivations for Eqn. (5.1) *and* (5.2)

$$\hat{C}_t = \Big(\Sigma_0^{-1} + \frac{1}{\sigma^2}\sum_{\tau=1}^{t} f^{\tau}_{A_\tau}(f^{\tau}_{A_\tau})^\top\Big)^{-1}; \tag{A.1}$$

$$\hat{\mu}_t = \hat{C}_t\Big(\Sigma_0^{-1}\mu_0 + \frac{1}{\sigma^2}\sum_{\tau=1}^{t} y(A_\tau) f^{\tau}_{A_\tau}\Big). \tag{A.2}$$

Proof. We denote $x_\tau = f_{A_\tau}$ for all $\tau = 1, \ldots, t$ and $y_\tau = y(A_\tau)$. Consider the following:

$$
\begin{aligned}
& p(w|\mathcal{D}_t) \propto p(\mathcal{D}_t|w)p(w) \propto \prod_{\tau=1}^{t} p(x_\tau, y_\tau|w)p(w) \\
& \propto \prod_{\tau=1}^{t} \exp\{-\frac{1}{2\sigma^2}(y_\tau - w^\top x_\tau)^2\} \\
& \exp\{-\frac{1}{2}(w-\mu_0)^\top \Sigma_0^{-1}(w-\mu_0)\} \\
& \propto \exp\{\sum_{\tau=1}^{t} -\frac{1}{2\sigma^2}(y_\tau - w^\top x_\tau)^2\} \\
& \exp\{-\frac{1}{2}(w-\mu_0)^\top \Sigma_0^{-1}(w-\mu_0)\} \\
& \propto \exp\{-\frac{1}{2}[\frac{1}{\sigma^2}\sum_{\tau=1}^{t} w^\top x_\tau x_\tau^\top w - \frac{2}{\sigma^2}\sum_{\tau=1}^{t} y_\tau x_\tau^\top w + \frac{1}{\sigma^2}\sum_{\tau=1}^{t} y_\tau^2 \\
& + w^\top \Sigma_0^{-1} w - 2\mu_0^\top \Sigma_0^{-1} w + \mu_0^\top \Sigma_0^{-1}\mu_0] \\
& \propto \exp\{-\frac{1}{2}[w^\top (\frac{1}{\sigma^2}\sum_{\tau=1}^{t} x_\tau x_\tau^\top + \Sigma_0^{-1})w \\
& - (\frac{2}{\sigma^2}\sum_{\tau=1}^{t} y_\tau x_\tau^\top + 2\mu_0^\top \Sigma_0^{-1})w + \frac{1}{\sigma^2}\sum_{\tau=1}^{t} y_\tau^2 + \mu_0^\top \Sigma_0^{-1}\mu_0]\} \\
& \propto \exp\{-\frac{1}{2}[w - (\frac{1}{\sigma^2}\sum_{\tau=1}^{t} x_\tau x_\tau^\top + \Sigma_0^{-1})^{-1} \\
& (\frac{1}{\sigma^2}\sum_{\tau=1}^{t} y_\tau x_\tau + \mu_0^\top \Sigma_0^{-1})]^\top (\frac{1}{\sigma^2}\sum_{\tau=1}^{t} x_\tau x_\tau^\top + \Sigma_0^{-1}) \\
& [w - (\frac{1}{\sigma^2}\sum_{\tau=1}^{t} x_\tau x_\tau^\top + \Sigma_0^{-1})^{-1}(\frac{1}{\sigma^2}\sum_{\tau=1}^{t} y_\tau x_\tau + \mu_0^\top \Sigma_0^{-1})]\} \\
& \propto \exp\{-\frac{1}{2}(x-\hat{\mu}_t)^\top \hat{C}_t^{-1}(x-\hat{\mu}_t)\}
\end{aligned}
$$

□

A.3 Proof for Lemma 1

Lemma. *Let* $f_i(A) = \big(m_i(A)\big)^\beta$ *with* $0 < \beta \leq 1$ *for* $i = 1, \ldots, d$. *Given* i *and* j, *let* $r_{i,j}(a) = \frac{m_i(a)}{m_j(a)}$ *(assuming* $m_i(a), m_j(a) > 0, \forall a \in V$*). When* $0 < \beta < 1/2$, $f_i(A)f_j(A)$ *is*

guaranteed to be monotone submodular, if $r_{i,j}(a) \in [\theta(\beta), 1/\theta(\beta)], \forall a \in V$, *where* $\theta(\beta) = -\frac{\sqrt{1-2\beta}}{\beta} - 1 + \frac{1}{\beta} \in (0,1)$. *When* $1/2 < \beta \leq 1$, $f_i(A)f_j(A)$ *is guaranteed to be monotone supermodular if* $r_{i,j}(a) \in [\theta'(\beta), 1/\theta'(\beta)], \forall a \in V$, *where* $\theta'(\beta) = \sqrt{\frac{\beta-\sqrt{2\beta-1}}{1-\beta}} \in (0,1)$.

Proof. For simplicity of notations, we write β as c, $r_{i,j}$ as r, $m_i(A), m_j(A)$ as $m_1(A), m_2(A)$, $\theta(\beta)$ as $\theta_-(c)$, $1/\theta(\beta)$ as $\theta_+(c)$, and the product $f_i(A)f_j(A)$ as $p(A)$.

We first prove the first half of the Theorem, i.e., given the condition $r(a) \in [\theta_-(c), \theta_+(c)]$ for all $a \in A$ and $0 \leq c < 1/2$, $p(A)$ is monotone submodular. The monotonicity of the set function $p(A)$ can be trivially verified. To prove submodularity of $p(A)$, we consider the problem in the continuous space. First we denote $f(x) = x^c$, and $g(x,y) = f(x)f(y)$. Both f and g are defined over the continuous space. The marginal gain $p(a|A)$ can be equivalently written as $g(m_1(A+a), m_2(A+a)) - g(m_1(A), m_2(A))$. Similarly, we can write $p(a|B) = g(m_1(B+a), m_2(B+a)) - g(m_1(B), m_2(B))$. To prove submodularity of $p(A)$, we need to equivalently show that $p(a|A) \geq p(a|B), \forall a \in V, A \subseteq B \subseteq V$. It is easy to see that the marginal gain $p(a|A)$ can be seen as the integral of the directional derivative of $g(x,y)$ in the direction $u = [m_1(a), m_2(a)]^\top$ from the point $z_1 = [m_1(A), m_2(A)]^\top$ to the point $z_1' = [m_1(A+a), m_2(A+a)]^\top$. Mathematically, it can be written as follows:

$$p(a|A) = \int_{s \in C_1} \nabla g(s) ds, \tag{A.3}$$

where C_1 is the line segment from the point z_1 to z_1'. Similarly, the marginal gain $p(a|B)$ can be interpreted as the integral of the derivative of the function $g(x,y)$ in the same direction $u = [m_1(a), m_2(a)]^\top$ from point $z_2 = [m_1(B), m_2(B)]^\top$ to the point $z_2' = [m_1(B+a), m_2(B+1)]^\top$. Let $v = z_2 - z_1$ be the vector difference between point z_2 and z_1. Then, we have the following:

$$p(a|B) = \int_{s \in C_2} \nabla g(s) ds = \int_{s \in C_1} \nabla g(v+s) ds, \tag{A.4}$$

where C_2 is the line segment from z_2 to z_2'. To establish that $p(a|A) \geq p(a|B), \forall a \in V, A \subseteq B \subseteq V$, it is sufficient to show the following:

$$\nabla g(s) ds - \nabla g(v+s) ds \geq 0, \forall s \in C_1, \tag{A.5}$$

Since $\nabla g(v+s)ds - \nabla g(s)ds = \int_{w \in C_3} \nabla(\nabla g(s+w)ds)dw$, where C_3 is the line segment from the origin $[0,0]^\top$ and v. Therefore, the above condition can be shown if the following condition holds:

$$\nabla(\nabla g(s+w)ds)dw \leq 0, \forall s \in C_1, w \in C_3. \tag{A.6}$$

We denote $\lambda = \frac{dw_2}{dw_1}$, $\beta = \frac{ds_2}{ds_1}$, and $\gamma = \frac{y}{x}$. Consider the following:

$$\begin{aligned}
&\nabla(\nabla g(x,y)ds)dw = \nabla(\frac{\partial g(x,y)}{\partial x}ds_1 + \frac{\partial g(x,y)}{\partial y}ds_2)dw \\
&= \Big[(\frac{\partial^2 g(x,y)}{\partial x^2}ds_1 + \frac{\partial^2 g(x,y)}{\partial x \partial y}ds_2)d_{w_1} \\
&+ (\frac{\partial^2 g(x,y)}{\partial x \partial y}ds_1 + \frac{\partial^2 g(x,y)}{\partial^2 y}ds_2)d_{w_2}\Big] \\
&= \Big[d_{w_1}d_{s_1}\Big(\frac{\partial^2 g(x,y)}{\partial x^2} + (\lambda+\beta)\frac{\partial^2 g(x,y)}{\partial x \partial y} + \lambda\beta\frac{\partial^2 g(x,y)}{\partial y^2}\Big)\Big]
\end{aligned}$$

Next, we plug in the definition of $g(x,y)$ and its partial derivatives into the above equation and obtain the following:

$$\begin{aligned}
&\nabla(\nabla g(x,y)ds)dw \\
&= d_{w_1}d_{s_1}\Big(c(c-1)x^{c-2}y^c + (\lambda+\beta)c^2x^{c-1}y^{c-1} \\
&+ c(c-1)x^c y^{c-2}\lambda\beta\Big) \\
&= cd_{w_1}d_{s_1}x^{c-1}y^{c-1}\Big((c-1)\gamma + (\lambda+\beta)c + (c-1)\frac{1}{\gamma}\lambda\beta\Big) \\
&\leq cd_{w_1}d_{s_1}x^{c-1}y^{c-1}\Big(2(c-1)\sqrt{\lambda\beta} + (\lambda+\beta)c\Big) \\
&\leq cd_{w_1}d_{s_1}x^{c-1}y^{c-1}\beta\Big(c\frac{\lambda}{\beta} + 2(c-1)\sqrt{\frac{\lambda}{\beta}} + c\Big)
\end{aligned}$$

Let $\rho = \sqrt{\frac{\lambda}{\beta}}$. Given that $r(a) = \frac{m_1(a)}{m_2(a)} \in [\theta_-(c), \theta_+(c)]$, we can derive that $\lambda, \beta \in [\theta_-(c), \theta_+(c)]$. Since $\theta_-(c)\theta_+(c) = 1$, we can also show that $\rho \in [\theta_-(c), \theta_+(c)]$. To show Eqn. (A.6), it is sufficient to prove $c\rho^2 + 2(c-1)\rho + c \leq 0$ for $\rho \in [\theta_-(c), \theta_+(c)]$. This is easy to see since $\theta_-(c)$ and $\theta_+(c)$ are the two roots for the equation $c\rho^2 + 2(c-1)\rho + c = 0$.

To prove the supermodularity of $p(A)$ given the condition $r(a) \in [\theta'_-(c), \theta'_+(c)]$ and $1/2 < c \leq 1$. We utilize the same proof techniques as above. In fact, it is sufficient to show the following:

$$\nabla(\nabla g(s+w)ds)dw \geq 0, \forall s \in C_1, w \in C_3. \tag{A.7}$$

This condition can be sufficiently proved if the following holds:

$$\nabla(\nabla g(x,y)ds)dw \tag{A.8}$$

$$= cd_{w_1}d_{s_1}x^{c-1}y^{c-1}\Big((c-1)\gamma + (\lambda+\beta)c + (c-1)\frac{1}{\gamma}\lambda\beta\Big) \tag{A.9}$$

$$\geq 0 \tag{A.10}$$

Consider the following:

$$(c-1)\gamma + (\lambda+\beta)c + (c-1)\frac{1}{\gamma}\lambda\beta \tag{A.11}$$

$$\geq (c-1)\gamma + 2\sqrt{\lambda\beta}c + (c-1)\frac{1}{\gamma}\lambda\beta \tag{A.12}$$

$$= \sqrt{\lambda\beta}\Big((c-1)\frac{\gamma}{\sqrt{\lambda\beta}} + 2c + (c-1)\frac{\sqrt{\lambda\beta}}{\gamma}\Big) \tag{A.13}$$

Suppose that $r(a) = \frac{m_1(a)}{m_2(a)} \in [s, \frac{1}{s}]$ with $0 < s \leq 1$, we can derive that $\lambda, \beta, \gamma \in [s, 1/s]$. Let $\alpha = \frac{\gamma}{\sqrt{\lambda\beta}}$. We can also show that $\alpha \in [s^2, 1/s^2]$. We then have the following:

$$(c-1)\gamma + (\lambda+\beta)c + (c-1)\frac{1}{\gamma}\lambda\beta \tag{A.14}$$

$$\geq \sqrt{\lambda\beta}\Big((c-1)\alpha + (c-1)\frac{1}{\alpha} + 2c\Big) \tag{A.15}$$

$$\geq \sqrt{\lambda\beta}\Big((c-1)(s^2 + (c-1)\frac{1}{s^2} + 2c\Big) \tag{A.16}$$

$$= \sqrt{\lambda\beta}\frac{1}{s^2}\Big((c-1)s^4 + 2cs^2 + (c-1)\Big) \tag{A.17}$$

Given that the equation $(c-1)s^4+2cs^2+(c-1)$ has a root $\hat{s} = \sqrt{\frac{c-\sqrt{2c-1}}{1-c}}$ and that $0 \leq \hat{s} \leq 1$, it can be shown that if s is chosen such that $s \geq \hat{s}$, the above becomes always lower bounded by 0. Therefore, the sufficient condition for $p(A)$ to be monotone supermodular is to ensure that $r(a) \in [\sqrt{\frac{c-\sqrt{2c-1}}{1-c}}, \sqrt{\frac{c+\sqrt{2c-1}}{1-c}}]$ □

A.4 Proof for Lemma 2

Lemma. *Given i, j, $\gamma > 0$ and $\delta > \frac{e-1}{\gamma}$, let $f_i(A) = \log(1 + \gamma m_i(A))$ for $i = 1, \ldots, d$, and $r_{i,j}(a) = \frac{m_i(a)}{m_j(a)}, \forall a \in V$. $f_i(A)f_j(A)$ is guaranteed to be monotone submodular if $m_i(a) \geq \delta, \forall i = 1, \ldots, d, a \in V$ and $r_{i,j}(a) \in [\theta(\gamma, \delta), 1/\theta(\gamma, \delta)], \forall a \in V$, where $\theta(\gamma, \delta) = \log(1 + \gamma\delta) - \sqrt{\log^2(1 + \gamma\delta) - 1}$.*

Proof. For simplicity of notations, we write $r_{i,j}$ as r, $m_i(A), m_j(A)$ as $m_1(A), m_2(A)$, and the product $f_i(A)f_j(A)$ as $p(A)$.

The Lemma can be restarted as: given the condition of $r(a) \in [\theta(\gamma, \delta), 1/\theta(\gamma, \delta)], \forall a \in A$ and $m_1(a), m_2(a) \geq \delta, \forall a \in V$, $p(A)$ is monotone submodular. The monotonicity of the set function $p(A)$ can be trivially verified. To prove submodularity of $p(A)$, we adopt the same strategy for proving Lemma 1. We consider the problem in the continuous space and use the same notation as in the proof for Lemma 1. Let $g(x, y) = \log(1 + \gamma x)\log(1 + \gamma y)$. Let $\lambda = \frac{dw_2}{dw_1}$, and $\beta = \frac{ds_2}{ds_1}$. To prove the submodularity of $p(A)$, it is sufficient to show the following holds given the condition on the function:

$$\Big(\frac{\partial^2 g(x, y)}{\partial x^2} + (\lambda + \beta)\frac{\partial^2 g(x, y)}{\partial x \partial y} + \lambda\beta\frac{\partial^2 g(x, y)}{\partial y^2}\Big) \leq 0.$$

Let $\delta' = \log(1 + \gamma\delta)$. Next, we plug in the definition of $g(x, y)$ and its partial derivatives into the above equation and obtain the following:

$$\begin{aligned}
&\Big(\frac{\partial^2 g(x, y)}{\partial x^2} + (\lambda + \beta)\frac{\partial^2 g(x, y)}{\partial x \partial y} + \lambda\beta\frac{\partial^2 g(x, y)}{\partial y^2}\Big) \\
&= -\frac{\gamma^2}{(1 + \gamma x)^2}\log(1 + \gamma y) + (\lambda + \beta)\frac{\gamma^2}{(1 + \gamma x)(1 + \gamma y)} \\
&- \lambda\beta\frac{\gamma^2}{(1 + \gamma y)^2}\log(1 + \gamma x) \\
&\leq -\frac{\gamma^2}{(1 + \gamma x)^2}\delta' + (\lambda + \beta)\frac{\gamma^2}{(1 + \gamma x)(1 + \gamma y)} \\
&- \lambda\beta\frac{\gamma^2}{(1 + \gamma y)^2}\delta' \\
&= \frac{\gamma^2}{(1 + \gamma x)(1 + \gamma y)}\Big[-\frac{1 + \gamma y}{1 + \gamma x}\delta' - \lambda\beta\frac{1 + \gamma x}{1 + \gamma y}\delta' + (\lambda + \beta)\Big]
\end{aligned}$$

$$\leq \frac{\gamma^2}{(1+\gamma x)(1+\gamma y)}[-2\delta'\sqrt{\lambda\beta}+(\lambda+\beta)]$$
$$= \frac{\gamma^2\beta}{(1+\gamma x)(1+\gamma y)}[\frac{\lambda}{\beta}-2\delta'\sqrt{\frac{\lambda}{\beta}}+1)]$$

Let $\rho = \sqrt{\frac{\lambda}{\beta}}$. Given that $r(a) = \frac{m_1(a)}{m_2(a)} \in [\theta(\gamma,\delta), 1/\theta(\gamma,\delta)]$, we can derive that $\lambda, \beta \in [\theta(\gamma,\delta), 1/\theta(\gamma,\delta)]$. We can also show that $\rho \in [\theta(\gamma,\delta), 1/\theta(\gamma,\delta)]$. Next, we only need to prove that $\rho^2 - 2\delta'\rho + 1 \leq 0$ for $\rho \in [\theta(\gamma,\delta), 1/\theta(\gamma,\delta)]$. This is easy to see since $\theta(\gamma,\delta)$ and $1/\theta(\gamma,\delta)$ are the two roots for the equation $\rho^2 - 2\delta'\rho + 1 = 0$.

□

A.5 Proof for Lemma 3

Lemma. *Let $f_i(A) = 1 - \prod_{a\in A}(1-p_i(a))$ with $p_i(a) \in [0,1]$ for $i = 1,\ldots,d$. Given any i and j, it holds that $f_i(A)f_j(A) = g(A) - h(A)$, where both $h(A) = 1 - \prod_{a\in A}(1-p_i(a) - p_j(a) + p_i(a)p_j(a))$ and $g(A) = f_i(A) + f_j(A)$ are monotone submodular.*

Proof. Consider the following for the product of the two functions:

$$f_i(A)f_j(A) = \Big(1 - \prod_{a\in A}(1-p_i(a))\Big)\Big(1 - \prod_{a\in A}(1-p_j(a))\Big) \tag{A.18}$$
$$= \Big(1 - \prod_{a\in A}(1-p_i(a))\Big) + \Big(1 - \prod_{a\in A}(1-p_j(a))\Big) \tag{A.19}$$
$$- \Big(1 - \prod_{a\in A}\prod_{a'\in A}(1 - p_j(a')(1-p_i(a)))\Big) \tag{A.20}$$
$$= \Big(1 - \prod_{a\in A}(1-p_i(a))\Big) + \Big(1 - \prod_{a\in A}(1-p_j(a))\Big) \tag{A.21}$$
$$- \Big(1 - \prod_{a\in A}(1 - p_i(a) - p_j(a) + p_i(a)p_j(a))\Big) \tag{A.22}$$
$$= g(A) - h(A) \tag{A.23}$$

It's easy to verify that $g(A)$ is monotone submodular. The remaining question is to establish the submodularity of $h(A)$. We prove the submodularity by using the definition of

submodularity. The marginal gain of an item a to a subset $A \subseteq V$ can be written as follows:

$$h(a'|A) = h(a' + A) - h(A) \tag{A.24}$$

$$= - \prod_{a \in (A+a')} (1 - p_i(a) - p_j(a) + p_i(a)p_j(a)) \tag{A.25}$$

$$+ \prod_{a \in A} (1 - p_i(a) - p_j(a) + p_i(a)p_j(a)) \tag{A.26}$$

$$= \Big(\prod_{a \in A} (1 - p_i(a) - p_j(a) + p_i(a)p_j(a)) \Big) \tag{A.27}$$

$$\Big(p_i(a') + p_j(a') - p_i(a')p_j(a') \Big) \tag{A.28}$$

Since $0 \leq p_i(a), p_j(a) \leq 1$, it is easy to see that $p_i(a') + p_j(a') - p_i(a')p_j(a') \geq 0$, therefore, $h(A)$ is monotone. Since the first term $\big(\prod_{a \in A} (1 - p_i(a) - p_j(a) + p_i(a)p_j(a)) \big)$ becomes smaller as the subset A grows, $h(A)$ also satisfies submodularity. □

A.6 Proof of Lemma 5

Lemma. LazyGreed *applied on the reduced ground set* $\hat{V} = \{j \in V | f(j) \geq f(u_\ell | V \setminus u_\ell)\}$ *is equivalent to that applied on the ground set* V*.*

Proof. Let $s_1, \ldots, s_\ell$ be the sequence of items selected by an instance of the LazyGreed. We denote that $S_i = \{s_1, \ldots, s_i\}$, such that $S_1 \subset S_2 \subset \cdots \subset S_\ell$. Consider the sequence of items $\{u_1, \ldots, u_n\}$ ordered non-increasingly in terms of their gain conditioned on all other items, i.e., $f(u_1 | V \setminus u_1) \geq \cdots \geq f(u_n | V \setminus u_n)$. Without loss of generality, we assume that $S_\ell \neq \{u_1, \ldots, u_\ell\}$, otherwise, the Lemma follows trivially. Next, we show the following:

$$f(s_i | S_{i-1}) \geq f(u_i | V \setminus u_i), \forall i = 1, \ldots, \ell. \tag{A.29}$$

We consider two cases:

(1) $u_i \notin S_{i-1}$

Consider the following:

$$f(s_i | S_{i-1}) = \max_{v \in V} f(v | S_{i-1})$$

$$\geq f(u_i|S_{i-1})$$
$$\geq f(u_i|V \setminus u_i)$$

(2) $u_i \in S_{i-1}$

Since S_{i-1} is of size $i-1$ and contains an item u_i, S_{i-1} cannot include all items in the set $\{u_1, \ldots, u_{i-1}\}$. Therefore, there exists an index $t \leq i-1$ such that $u_t \notin S_{i-1}$. We have the following:

$$f(s_i|S_{i-1}) = \max_{v \in V} f(v|S_{i-1})$$
$$\geq f(u_t|S_{i-1})$$
$$\geq f(u_t|V \setminus u_t)$$
$$\geq f(u_i|V \setminus u_i)$$

For items $j \in V \setminus \hat{V}$, the singleton gain can be bounded as the following:

$$f(j) < f(u_\ell|V \setminus u_\ell) \leq f(s_\ell|S_{\ell-1})$$

The singleton gain of the items in the removed data set is strictly less than smallest the marginal gain of the selected item by the greedy algorithm, hence, it is guaranteed that the greedy selection does not pick items from $V \setminus \hat{V}$, hence the pruning procedure does not affect the performance of the standard greedy procedure. □

A.7 Proof of Theorem 13

Theorem. *Given a target submodular function f with total curvature κ_f, an instance of* MultGreed *with greedy ratio α is guaranteed to obtain a set S_ℓ s.t.*

$$\frac{f(S_\ell)}{f(S^{OPT})} \geq \frac{1}{\kappa_f}\big(1 - (1 - \frac{1}{\alpha\ell})^{\ell\kappa_f}\big) \geq \frac{1}{\kappa_f}(1 - e^{-\frac{\kappa_f}{\alpha}})$$
$$\geq (1 - e^{-\frac{1}{\alpha}})$$

Conversely, for any value of $\alpha \geq 1$ and $\kappa_f \in [0, 1]$, there exists a submodular f with the total curvature κ_f, on which an instance of MultGreed *with the greedy ratio α achieves an approximation factor $\frac{1}{\kappa_f}\big(1 - (1 - \frac{1}{\alpha\ell})^{\ell\kappa_f}\big)$.*

Proof. We employ similar proof techniques from [Conforti and Cornuejols, 1984], where they show the curvature dependent bound for Problem 6.2. We generalize their results by introducing the greedy ratio in the bound.

To prove the theorem, we need the following two Lemmas.

Lemma 12.

$$\prod_{i=1}^{n}(1-\frac{1}{x_i}) \leq (1-\frac{1}{\bar{x}_h})^n \tag{A.30}$$

where $x_i \geq 1, \forall i = 1, \ldots, n$, *and* $\bar{x}_h$ *is harmonic mean of* x_i*'s, i.e.,* $\bar{x}_h = \frac{n}{\sum_{i=1}^{n}\frac{1}{x_i}}$.

Proof. Consider the fact that geometric mean of any non-negative vector lower bounds arithmetic mean of the same vector, namely,

$$(\prod_{i=1}^{n} z_i)^{1/n} \leq \frac{\sum_{i=1}^{n} z_i}{n} \tag{A.31}$$

where $z_i \geq 0, \forall i$.

Let $z_i = 1 - \frac{1}{x_i}$, where $x_i \geq 1$, we have

$$(\prod_{i=1}^{n}(1-\frac{1}{x_i}))^{1/n} \leq \frac{\sum_{i=1}^{n}(1-\frac{1}{x_i})}{n} \tag{A.32}$$

Then,

$$\prod_{i=1}^{n}(1-\frac{1}{x_i}) \leq (\frac{\sum_{i=1}^{n}(1-\frac{1}{x_i})}{n})^n \tag{A.33}$$

Rearrange the term in the R.H.S,

$$\frac{\sum_{i=1}^{n}(1-\frac{1}{x_i})}{n} = 1 - \frac{\sum_{i=1}^{n}\frac{1}{x_i}}{n} \tag{A.34}$$

$$= 1 - \frac{1}{\frac{n}{\sum_{i=1}^{n}\frac{1}{x_i}}} \tag{A.35}$$

$$= 1 - \frac{1}{\bar{x}_h} \tag{A.36}$$

□

Lemma 13. *For $t = 0, \ldots, \ell - 1$,*

$$\sum_{i:s_i \in S^t \cap S^{OPT}} f(s_i|S^{i-1}) + \kappa_f \sum_{i:s_i \in S^t \setminus S^{OPT}} f(s_i|S^{i-1}) +$$
$$\alpha_{t+1}(\ell - |S^t \cap S^{OPT}|) f(s_{t+1}|S^t) \geq f(S^{OPT}),$$

where $S^i = \{s_1, \ldots, s_i\}$, ℓ is the size constraint, κ_f is the total curvature of f, and α_i is the individual greedy ratio for iteration i.

Proof. By submodularity and definition of the greedy ratio, we have the following inequality:

$$f(S^{\mathrm{OPT}} \cup S^t) \leq f(S^t) + \sum_{u \in S^{\mathrm{OPT}} \setminus S^t} f(u|S^t) \tag{A.37}$$
$$\leq f(S^t) + \sum_{u \in S^{\mathrm{OPT}} \setminus S^t} \alpha_{t+1} f(s_{t+1}|S^t) \tag{A.38}$$
$$= f(S^t) + |S^{\mathrm{OPT}} \setminus S^t| \alpha_{t+1} f(s_{t+1}|S^t) \tag{A.39}$$

By the definition of curvature, we have

$$f(S^{\mathrm{OPT}} \cup S^t) \tag{A.40}$$
$$= f(S^{\mathrm{OPT}}) + \sum_{i:s_i \in S^t \setminus S^{\mathrm{OPT}}} f(s_i|S^{\mathrm{OPT}} \cup S^{i-1}) \tag{A.41}$$
$$\geq f(S^{\mathrm{OPT}}) + (1 - \kappa_f) \sum_{i:s_i \in S^t \setminus S^{\mathrm{OPT}}} f(s_i|S^{i-1}) \tag{A.42}$$

Combining inequalities A.39 and A.42, we can get the following bound:

$$f(S^{\mathrm{OPT}}) + (1 - \kappa_f) \sum_{i:s_i \in S^t \setminus S^{\mathrm{OPT}}} f(s_i|S^{i-1})$$
$$\leq f(S^t) + |S^{\mathrm{OPT}} \setminus S^t| \alpha_{t+1} f(s_{t+1}|S^t)$$

Then,

$$f(S^{\mathrm{OPT}}) + (1 - \kappa_f) \sum_{i:s_i \in S^t \setminus S^{\mathrm{OPT}}} f(s_i|S^{i-1})$$

$$\leq \sum_{s_i \in S^t} f(s_i|S^{i-1}) + (\ell - |S^t \cap S^{\text{OPT}}|)\alpha_{t+1} f(s_{t+1}|S^t)$$

□

Consider the family $\mathcal{F}_{f,\ell,\kappa}$ of all instances of Problem 6.2. For notation simplicity we write $\mathcal{F} = \mathcal{F}_{\{,\ell,\kappa}$. Given a problem instance, we assume that the multi-stage greedy algorithm MultGreed returns a subset S^ℓ and $|S^\ell \cap S^{\text{OPT}}| = m$. For $0 \leq m \leq \ell$, let $1 \leq i_1 < i_2 < , \ldots, < i_m \leq \ell$ be a sequence of integers such that $S^\ell = \{s_1, s_2, \ldots, s_\ell\}$ has the elements $s_{i_1}, \ldots, s_{i_m}$ in common with the optimal solution S^{OPT}. Note that when $m = 0$, the set of common elements is empty. By Lemma 13, it's easy to verify that $\frac{f(S^\ell)}{f(S^{\text{OPT}})} \geq B(i_1, \ldots, i_m)$, where $B(i_1, \ldots, i_m)$ is the solution to the following optimization problem:

$$\min_{\rho_1,\ldots,\rho_\ell} \quad \mathbb{1}^T \rho \tag{A.43}$$

$$\text{subject to} \quad \rho_i \geq 0, i = 1, \ldots \ell \tag{A.44}$$

$$A(i_1, \ldots, i_m)\rho \geq 1 \tag{A.45}$$

Note the matrix $A(i_1, \ldots, i_m)$ is defined the same as the matrix used in the proof of Lemma 5.1 in [Conforti and Cornuejols, 1984], except that we have all the diagonal entries of the matrix scaled by α_i for each i. Since the matrix $A(i_1, \ldots, i_m)$ is full rank, the polyhedron given by $A(i_1, \ldots, i_m)\rho \geq 1$ has only one vertex, and solving $A(i_1, \ldots, i_m)\rho = 1$ yields the vertex $\rho^* \geq 0$. Note that the optimum of an LP can always be obtained at a vertex,the only feasible vertex ρ^* is guaranteed to be among the optimum. Therefore ρ^* is also the solution to the linear program. Hence, $B(i_1, \ldots, i_m) = \sum_{i=1}^{\ell} \rho_i^*$.

Next,we are going to show that $B(i_1, \ldots, i_m) \geq B(\emptyset)$. We start with the case where $m = 1$, and assume that $S^\ell \cap S^{\text{OPT}} = \{s_p\}, 1 \leq p \leq \ell$. We denote the solution to the problem $B(p)$ as ρ', and the solution to $B(\emptyset)$ as ρ. Solving for ρ and ρ' yields the following:

$$\rho_i = \frac{1}{\alpha_i \ell} \prod_{j=1}^{i-1} (1 - \frac{\kappa}{\alpha_j \ell}), \forall i = 1, \ldots, \ell$$

$$\rho'_i = \rho_i, \forall i = 1, \ldots, p;$$

$$\rho'_p = \frac{1}{(\ell-1)\alpha_p}(1-\frac{1}{\ell\alpha_p})\prod_{j=1}^{p-1}(1-\frac{\kappa}{\ell\alpha_j})$$

$$\rho'_i = \frac{1}{(\ell-1)\alpha_i}(1-\frac{1}{\ell\alpha_p})\prod_{j=1}^{p-1}(1-\frac{\kappa}{\ell\alpha_j})\prod_{j=p=1}^{i-1}(1-\frac{\kappa}{(\ell-1)\alpha_j}),$$

for $i = p+1,\ldots,\ell$;

To show that $B(p) \geq B(\emptyset)$, it suffices to show that $\rho_i \leq \rho'_i$ for any i, equivalently $\frac{\rho_i}{\rho'_i} \leq 1$. Consider the following for $i = p,\ldots,\ell$

$$\frac{\rho_i}{\rho'_i} = \frac{\ell-1}{\ell}\frac{1-\frac{\kappa}{\ell\alpha_p}}{1-\frac{1}{\ell\alpha_p}}\prod_{j=p+1}^{i-1}\frac{1-\frac{\kappa}{\ell\alpha_j}}{1-\frac{\kappa}{(\ell-1)\alpha_j}} \tag{A.46}$$

$$\leq \frac{\ell-1}{\ell}\frac{1-\frac{\kappa}{\ell}}{1-\frac{1}{\ell}}\prod_{j=p+1}^{i-1}\frac{1-\frac{\kappa}{\ell}}{1-\frac{\kappa}{\ell-1}} \tag{A.47}$$

$$\leq \frac{(1-\frac{\kappa}{\ell})^\ell - 1}{(1-\frac{\kappa}{(\ell-1)})^\ell - 2} \tag{A.48}$$

$$\leq 1 \tag{A.49}$$

The first inequality follows since $\frac{1-\frac{\kappa}{\ell\alpha}}{1-\frac{\kappa}{(\ell-1)\alpha}}$ is monotonically non-increasing with α, hence, its maximum is attained when $\alpha = 1$. The last inequality follows from the fact that $(1-\frac{\kappa}{x})^{x-1}$, where $\kappa \in [0,1]$, is monotonically non-increasing when $x \geq 1$. We have shown that $\rho_i \leq \rho'_i$ for any i, hence, $B(\emptyset) \leq B(p)$. We are going to show that $B(i_1,\ldots,i_m) \geq B(\emptyset)$ by induction. It's already proved that it holds when $m = 1$. Suppose $B(i_1,\ldots,i_q) \geq B(\emptyset)$ holds, we wish to show the following: $B(i_1,\ldots,i_q,i_{q+1}) \geq B(i_1,\ldots,i_q)$ for any $i_{q+1} \notin \{i_1,\ldots,i_q\}$. We denote the solution to problem $B(i_1,\ldots,i_q)$ as ρ^{B_q}, and the solution to problem $B(i_1,\ldots,i_{q+1})$ as $\rho^{B_{q+1}}$. It's easy to see that $\rho_i^{B_{q+1}} = \rho_i^{B_q}$, for $i = 1,\ldots,i_{q+1}$. Let $r = i_{q+1}$, then for $i = r+1,\ldots,\ell$, we have the following:

$$\frac{\rho_i^{B_q}}{\rho_i^{B_{q+1}}} = \frac{\frac{1}{(\ell-q)\alpha_i}}{\frac{1}{(\ell-1-1)\alpha_i}}\frac{(1-\frac{\kappa}{(\ell-q)\alpha_r})}{1-\frac{1}{(\ell-q)\alpha_r}}\prod_{j=r+1}^{i-1}\frac{1-\frac{\kappa}{(\ell-q)\alpha_j}}{1-\frac{\kappa}{(\ell-q-1)\alpha_j}} \tag{A.50}$$

$$\leq \frac{\ell-q-1}{\ell-q}\frac{1-\frac{\kappa}{\ell-q}}{1-\frac{1}{\ell-q}}(\frac{1-\frac{\kappa}{\ell-q}}{1-\frac{\kappa}{\ell-q-1}})^{i-1-r} \tag{A.51}$$

$$\leq \frac{(1-\frac{\kappa}{\ell-q})^{\ell-1-1}}{(1-\frac{\kappa}{\ell-q-1})^{\ell-q-2}} \tag{A.52}$$

$$\leq 1 \tag{A.53}$$

Therefore, we have $\rho_i^{B_q} \leq \rho_i^{B_{q+1}}$ for all i, thus $B(i_1, \ldots, i_q) \leq B(i_1, \ldots, i_{q+1})$.

Then, we have the following:

$$\begin{aligned}
\frac{f(S^\ell)}{f(S^{\text{OPT}})} &\geq B(i_1, \ldots, i_m) \\
&\geq B(\emptyset) \\
&= \frac{1}{\kappa}(1 - \prod_{i=1}^{\ell}(1 - \frac{\kappa}{\ell\alpha_i})) \\
&\geq \frac{1}{\kappa}(1 - e^{-\frac{\kappa}{\alpha}})
\end{aligned}$$

where α is the harmonic mean of α_i's.

To show that the approximation factor is tight. We only show for the case where $\kappa = 1$. Results could be generalized for any value of κ.

We use the similar tight instance construction as in [Nemhauser and Wolsey, 1978], where they show a class of instances of the greedy algorithm that achieves the approximate factor $(1-(1-\frac{1}{\ell})^\ell)$. In this case, we are going to show an instance of MultGreed that achieves the approximate factor $(1-(1-\frac{1}{\alpha}\ell)^\ell)$, where α is the greedy ratio of this instance of MultGreed. To this end, let's define a family of submodular functions that provide worst-case examples over all possible $\alpha \geq 1$ and ℓ. The (α, ℓ)th subfamily is specified by function g_α^ℓ. Notice that g_α^ℓ is a set function on a ground set V of cardinality n. For notation simplicity, we drop the dependency on n in the set function g_α^ℓ, although there is a different function for each n. We will show how to construct a submodular function for a combination of n, α, ℓ, on which MultGreed achieves the approximation factor $(1 - (1 - \frac{1}{\alpha\ell})^\ell)$.

Consider ground set $V = \{1, 2, \ldots, n\}$ that contains two types of elements *special* and *plain*. The subset M, with $|M| = \ell$, is the set of special elements and $V \setminus M$ is the set of plain elements. Value of $g_\alpha^\ell(S), S \subseteq V$, depends only on $|S|, |S \cap M|, \alpha$ and ℓ. For this

reason, we write $g^\ell_\alpha(S) = g^\ell_\alpha(|S \cap M|, |S|) = g^\ell_\alpha(i, j)$, where $i = |S \cap M|$ and $j = |S|, i = 0, \ldots, \min(j, \ell), j = 0, \ldots, n$. Let $g^\ell_\alpha(i, j)$ for general ℓ, α, i, j be defined as following:

$$g^\ell_\alpha(0, j) = \ell(1 - (1 - \frac{1}{\alpha\ell})^j), 0 \leq j \leq n; \tag{A.54}$$

$$g^\ell_\alpha(i, j) = g^\ell_\alpha(0, j - i) + i\frac{\ell - g^\ell_\alpha(0, j - i)}{\ell}, \tag{A.55}$$

$$\text{for } 1 \leq i \leq \ell, i \leq j \leq n; \tag{A.56}$$

The fact that g^ℓ_α is monotone submodular is proved by tedious enumeration of the definition of submodularity and monotonicity case by case. We leave out the proof here and defer readers to [Nemhauser and Wolsey, 1978] for detail. Consider an instance of MultGreed performed on g^ℓ_α, that only choose items from $V \setminus M$. Then, the greedy ratio would be:

$$\alpha_i = \max_{u \in V} \frac{f(u|S_{i-1})}{f(s_i|S_{i-1})} = \frac{g^\ell_\alpha(1, i) - g^\ell_\alpha(0, i - 1)}{g^\ell_\alpha(0, i) - g^\ell_\alpha(0, i - 1)} = \alpha \tag{A.57}$$

Hence, the harmonic mean of α_i's is equal to α. Output subset of MultGreed is a subset of $V \setminus M$ with cardinality ℓ, thus has function value $g^\ell_\alpha(0, \ell) = \ell(1 - (1 - \frac{1}{\alpha\ell})^\ell)$, while optimal subset that maximizes submodular function f under cardinality constraint ℓ should be M and has function value $g^\ell_\alpha(M) = g^\ell_\alpha(\ell, \ell) = \ell$. Approximation factor becomes $\frac{g^\ell_\alpha(0,\ell)}{g^\ell_\alpha(M)} = 1 - (1 - \frac{1}{\alpha\ell})^\ell$. Therefore, the bound in terms of the greedy ratio α is tight. □

A.8 Proof of Theorem 14

Theorem. *Given a target submodular function f, an instance of* MultGreed *with knapsack greedy ratio α is guaranteed to obtain a set S s.t.*

$$f(S) \geq \frac{1}{2}(1 - e^{-\frac{1}{\alpha}})f(S^{OPT}) \tag{A.58}$$

where $S^{OPT} \in \text{argmax}_{S \subseteq V, c(S) \leq B} f(S)$.

Proof. Following the similar proof techniques from [Nemhauser et al., 1978] along with the definition of the knapsack greedy ratio in Eqn 6.7, we have the following:

$$f(S^{\text{OPT}}) \leq \frac{\alpha_i B}{c(s_i)} f(s_i|S_{i-1}) + f(S_{i-1}), \forall i = 1, \ldots, N \tag{A.59}$$

Rearrange the inequality, we get the following:

$$f(S^{\mathrm{OPT}}) - f(S_i) \leq (1 - \frac{c(s_i)}{\alpha_i B})(f(S^{\mathrm{OPT}}) - f(S_{i-1}))$$

holds for $i = 1, \ldots, N$. Then, we can obtain the following:

$$f(S_N) \geq (1 - \prod_{i=1}^{N}(1 - \frac{c(s_i)}{\alpha_i B}))f(S^{\mathrm{OPT}})$$

By definition of the knapsack greedy ratio in Eqn 6.8 and Lemma 12, we obtain the following:

$$\begin{aligned} f(S_N) &\geq (1 - (1 - \frac{B'}{\alpha B N})^N)f(S^{\mathrm{OPT}}) \\ &\geq (1 - e^{-\frac{B'}{B\alpha}})f(S^{\mathrm{OPT}}) \\ &\geq (1 - e^{-\frac{1}{\alpha}})f(S^{\mathrm{OPT}}) \end{aligned}$$

Since S_N is not a feasible solution to Problem 6.6, but S_{N-1} is. The knapsack greedy algorithm compares the solution between S_{N-1} and the maximum single value $f(v^*)$, where $v^* \in \mathrm{argmax}_{u \in V; c(u) \leq B} f(u)$, and outputs the maximum of the two. Then, we can bound the output solution as follows:

$$\begin{aligned} \max\{f(v^*), f(S_{N-1})\} &\geq \frac{1}{2}(f(v^*) + f(S_{N-1})) \\ &\geq \frac{1}{2}(f(s_N) + f(S_{N-1})) \\ &\geq \frac{1}{2}f(S_N) \\ &\geq \frac{1}{2}(1 - e^{-\frac{1}{\alpha}})f(S^{\mathrm{OPT}}) \end{aligned}$$

□

A.9 Proof for Theorem 15

Theorem. *An instance of* MultGreed *that solves Problem 6.6 with the knapsack greedy ratio α returns a solution that can be transformed to a solution S for Problem 6.10 such that*

$$c(S) \leq c(S^{OPT}) \text{ and } f(S) \geq \frac{1}{2}(1 - e^{-\frac{1}{\alpha}})f(V), \tag{A.60}$$

where $S^{OPT} \in \mathrm{argmax}_{S \subseteq V, f(S) \geq f(V)} c(S)$

Proof. This Theorem immediately follows from Theorem 14 and Theorem 3.1 of [Iyer and Bilmes, 2013]. □

A.10 Proof of Lemma 7

Lemma. *Using the surrogate uniform mixture* $\hat{f}^{sub}$ *for stage* j *in* MultGreed *gives individual greedy ratios of*

$$1 \leq \alpha_i \leq \frac{1}{1-\epsilon}, \forall i \in I_j, \tag{A.61}$$

with probability $1-\delta$*, where* $\delta = (1 - 5n^{\ell} e^{-\frac{np(g^{\ell})^2\epsilon^2}{64\mathcal{B}^2}})$ *and* $g^{\ell} = \max_{u \in V \setminus S_{\ell-1}} f(u|S_{\ell-1}) > 0$.

Proof. We employ methods that are similarly presented in [Mirzasoleiman et al., 2013], where they show the theoretical guarantee of applying $\hat{f}^{\text{sub}}$ as a surrogate function for the class of decomposable submodular functions. For notation simplicity, we write $\hat{f}^{\text{sub}}$ as $\hat{f}$. We denote the ground set size to be $|V| = n$. The expected size of the subset $\mathcal{T}' \subseteq \mathcal{T}$ is np. We are going to show that $\hat{f}$ is close to the target function f for all possible subset S of size $|S| \leq \ell$ with high probability, where ℓ is the size constraint of the selection. Given a fixed $S \subseteq V$, we can treat all $f_i(S)$'s as independent random variables and, by assumption, they are bounded as $0 \leq f_i(S) \leq \mathcal{B}$.

By Chernoff bound, we can bound the probability $\Pr(|\mathcal{T}'| \leq \frac{1}{2}np) \leq e^{-\frac{np}{2}}$. By the Hoeffding inequality, we can bound the probability $\Pr(|\hat{f}(S) - \mu| \geq \epsilon) \leq 2e^{-\frac{2n\epsilon^2}{\mathcal{B}^2}}$ and $\Pr(|\hat{f}(S) - \mu| \geq \epsilon) \leq 2e^{-\frac{np\epsilon^2}{\mathcal{B}^2}}$, where μ is the mean of the random variable $f_i(S)$ for any i.

By union bound, we have the following for ϵ being small such that $\frac{\epsilon^2}{\mathcal{B}^2} \leq \frac{1}{2}$.

$$\begin{aligned}\Pr(|f(S) - \hat{f}(S)| \leq 2\epsilon) &\geq 1 - 2e^{-\frac{2n\epsilon^2}{\mathcal{B}^2}} - 2e^{-\frac{np\epsilon^2}{\mathcal{B}^2}} - e^{-\frac{np}{2}} \\ &\geq 1 - 5e^{-\frac{np\epsilon^2}{\mathcal{B}^2}}\end{aligned}$$

There are in total n^{ℓ} sets of size less or equal to ℓ. By the union bound again, we can have the following:

$$\Pr(|f(S) - \hat{f}(S)| \leq 2\epsilon, \forall S \subseteq V, |S| \leq \ell) \geq 1 - 5n^{\ell} e^{-\frac{np\epsilon^2}{\mathcal{B}^2}} \tag{A.62}$$

Notice that this bound is meaningful only when $1 - 5n^{\ell}e^{-\frac{np\epsilon^2}{\mathcal{B}^2}} > 0$, in other words, we can obtain meaningful theoretical guarantee when the ground set size n is sufficiently large.

Now, we can analyze the performance guarantee in terms of the greedy ratio. Continuing from the results in the inequality A.62 , we have that

$$|f(j|S \setminus j) - \hat{f}(j|S \setminus j)| \leq 4\epsilon$$

for any $j \in V$, and $|S| \leq \ell$, with probability $(1-5n^{\ell}e^{-\frac{np\epsilon^2}{\mathcal{B}^2}})$. Let the gain in the last iteration of the greedy algorithm to be g_G^{ℓ}, and assume that $g_G^{\ell} > 0$. Let t_i be the item that attains the maximum marginal gain by applying the target function f and s_i be the item selected by the surrogate function $\hat{f}$ for iteration i, then we have the following

$$\begin{aligned}\alpha_i = \frac{f(t_i|S_{i-1})}{f(s_i|S_{i-1})} &\leq \frac{f(t_i|S_{i-1})}{\hat{f}(s_i|S_{i-1}) - 4\epsilon} \leq \frac{f(t_i|S_{i-1})}{\hat{f}(t_i|S_{i-1}) - 4\epsilon} \\ &\leq \frac{f(t_i|S_{i-1})}{f(t_i|S_{i-1}) - 8\epsilon} = \frac{1}{1 - \frac{8\epsilon}{g_G^{\ell}}} = \frac{1}{1-\epsilon'}\end{aligned}$$

where $\epsilon' = \frac{8\epsilon}{g_G^{\ell}}$. To formalize the result in terms of the greedy ratio, we can claim that with probability $(1 - 5n^{\ell}e^{-\frac{np(g_G^{\ell})^2\epsilon'^2}{64\mathcal{B}^2}})$, we can bound the greedy ratio in each iteration as $\alpha_i \leq \frac{1}{1-\epsilon'}$. □

A.11 Proof of Lemma 8

Lemma. *Using the modular upper bound as a surrogate function, it holds that*

$$1 \leq \alpha_i \leq \frac{1}{1 - \kappa_f(S_{i-1})}, \forall i \in I_j$$

Proof. Let t_i be the item with maximum marginal gain in iteration i, i.e., $t_i \in \text{argmax}_{u \in V} f(u|S_{i-1})$ By definition of the greedy ratio, we have the following:

$$\begin{aligned}\alpha_i = \frac{f(t_i|S_{i-1})}{f(s_i|S_{i-1})} &\leq \frac{f(t_i)}{f(s_i|S_{i-1})} \leq \frac{f(s_i)}{f(s_i|S_{i-1})} \\ &\leq \frac{1}{1 - \kappa_f(S_{i-1})}\end{aligned}$$

The second inequality follows from that fact that s_i is the item greedily selected by the modular proxy $\hat{f}^{\text{mod}}$, therefore $f(s_i) = \max_{u \in V \setminus S_{i-1}} f(u) \geq f(t_i)$. The last inequality follows from the definition of the curvature. □

A.12 Proof of Lemma 9

Lemma. *For the facility location function, we have:*

$$\hat{f}_{fac}^{k\text{-}NNG}(S) = f_{fac}(S), \forall S \subseteq V \ s.t. \ |S| \geq m, \tag{A.63}$$

with probability at least $(1-\theta)$*, and the sparsity of the k-NNG being at least*

$$k = n[1 - (\frac{\theta}{n})^{\frac{1}{m}}]. \tag{A.64}$$

Proof. Let $\vec{\hat{w}}_i$ be the ith row vector obtained from the k-NNG approximation from the full graph. Then, $\vec{\hat{w}}_i$ is the approximate vector for $\vec{w}_i$ with only k largest values retained. The key observation for the facility location function is that $f_{\text{fac}}(S) = \hat{f}_{\text{fac}}^{\text{k-NNG}}(S)$, if the set S contains items that are among the top k values of the row vector $\vec{w}_i$ for all i, since $\max_{j \in S} \vec{w}_i(j) = \max_{j \in S} \vec{\hat{w}}_i(j)$, if S contains items that are among the top k values of $\vec{w}_i$.

For notation simplicity, we write $\hat{f}$ for $\hat{f}_{\text{fac}}^{\text{k-NNG}}$ and f for f_{fac}. For any item $t \in V$, we have the probability of $w_{i,t}$ not being among the top k elements of the row vector w_i as $\frac{n-k}{n}$, given the uniform distribution assumption.

By the independence assumption, the probability, for which a set S_m of size m contain at least one item among the top k elements for each row vector, can be then computed as $[1 - (\frac{n-k}{n})^m]^n$.

Let the probability that S_m covers among the top k elements of all row vectors be $1-\theta$. Then, we have the following:

$$[1 - (\frac{n-k}{n})^m]^n = 1 - \theta$$

Simplify the equation, we can get the following:

$$k = n[1 - (1 - (1-\theta)^{\frac{1}{n}})^{\frac{1}{m}}] \tag{A.65}$$

$$\approx n[1-(1-e^{-\frac{\theta}{n}})^{\frac{1}{m}}] \tag{A.66}$$

$$\approx n[1-(\frac{\theta}{n})^{\frac{1}{m}}] \tag{A.67}$$

The first approximation follows since $(1-\theta)^{\frac{1}{n}} \approx e^{-\frac{\theta}{n}}$, for θ being close to 0. The second approximation follows from that $e^{-\frac{\theta}{n}} \approx 1-\frac{\theta}{n}$, with $-\frac{\theta}{n} \approx 0$. □

A.13 Proof of Lemma 10

Lemma. *Given the saturated coverage function, an instance of* MultGreed *with the size constraints* $\ell_1 = \lfloor \frac{n\xi}{(1-\xi)\gamma+\xi} \rfloor$ *and* $\ell_2 = \max\{0, \ell-\ell_1\}$ *(where* $\gamma = \frac{\max_{u,v} w_{u,v}}{\min_{(u,v)\in E(G)} w_{u,v}}$*, assuming all extent graph edges are positively weighted) yields a solution with the individual greedy ratios*

$$\alpha_i = 1, \text{ for } i = 1, \ldots, \ell_1$$

And with probability $1-\delta$*,*

$$1 \leq \alpha_i \leq \frac{1}{1-\epsilon}, \text{ for } i = \ell_1+1 \ldots, \ell$$

where $\delta = (1 - 5n^{\ell} e^{-\frac{np(g^{\ell})^2\epsilon^2}{64B^2}})$ *and* $g^{\ell} = \max_{u\in V\setminus S_{\ell-1}} f(u|S_{\ell-1}) > 0$.

Proof. The bound of the individual greedy ratio α_i's for $i = \ell_1+1, \ldots, \ell$ can be immediately derived from Lemma 7. Therefore, it is left to show that $\alpha_i = 1$ for $i = 1, \ldots, \ell_1$, which, by Lemma 8, is equivalently as to show that $\kappa_f(S) = 0$ for $|S| \leq \ell_1$. For notation simplicity, we write f_{sat} as f. To show that $\kappa_f(S) = 0$, we can equivalently show that $f(S) = \sum_{j\in S} f(j)$ for any S of size $|S| \leq \lfloor \frac{n\xi}{(1-\xi)\gamma+\xi} \rfloor$.

Consider the following for any $S \subseteq V$ such that $|S| \leq \lfloor \frac{n\xi}{(1-\xi)\gamma+\xi} \rfloor$:

$$\begin{aligned}
\frac{\sum_{j\in S} w_{i,j}}{\sum_{j\in V} w_{i,j}} &= \frac{\sum_{j\in S} w_{i,j}}{\sum_{j\in S} w_{i,j} + \sum_{j\in V\setminus S} w_{i,j}} \\
&\leq \frac{|S|\rho_{\max}}{|S|\rho_{\max} + (n-|S|)\rho_{\min}} \\
&= \frac{|S|\gamma}{|S|\gamma + (n-|S|)}
\end{aligned}$$

$$\leq \frac{1}{1+\frac{n}{\gamma\frac{n\xi}{(1-\xi)\gamma+\xi}}-\frac{1}{\frac{n\xi}{(1-\xi)\gamma+\xi}}}$$
$$= \xi$$

Without loss of generality, we can assume that all $w_{i,j} > 0$ for any $j \in S$ and a given i, since the above holds as well for $S' = \{j \in S | w_{i,j} > 0, \forall i\}$. Therefore, we have that $\sum_{j\in S} w_{i,j} \leq \xi \sum_{j\in V} w_{i,j}$ for all i. From which, we conclude that $\kappa_f(S) = 0$, for any $|S| \leq \lfloor \frac{n\xi}{(1-\xi)\gamma+\xi} \rfloor$ □

A.14 Proof of Lemma 11

Lemma. *Given the feature based submodular function, an instance of* MultGreed *with the size constraints being ℓ_1 and ℓ_2, yields a solution with the individual greedy ratio bounded as:*

$$1 \leq \alpha_i \leq O(i^{1-a}), \text{ for } i = 1, \ldots, \ell_1$$

And with probability $1-\delta$,

$$1 \leq \alpha_i \leq \frac{1}{1-\epsilon}, \text{ for } i = \ell_1 + 1 \ldots, \ell$$

where $\delta = (1 - 5n^\ell e^{-\frac{np(g^\ell)^2\epsilon^2}{64B^2}})$ and $g^\ell = \max_{u\in V\setminus S_{\ell-1}} f(u|S_{\ell-1}) > 0$.

Proof. The bound of the individual greedy ratio α_i's for $i = \ell_1 + 1, \ldots, \ell$ can be immediately derived from Lemma 7. Therefore, it is left to show that $\alpha_i \leq O(i^{1-a})$ for $i = 1, \ldots, \ell_1$. For notation simplicity, we write f_{fea} as f. Let $\rho_{\min} = \min_{u\in\mathcal{F}, v\in V: c_u(v)>0} c_u(v)$ and $\rho_{\max} = \max_{u\in\mathcal{F}, v\in V} c_u(v)$. It suffices to show the following:

$$\alpha_i \leq \frac{1}{(1+(i-1)\gamma)^a - ((i-1)\gamma)^a},$$

where $\gamma = \frac{\rho_{\max}}{\rho_{\min}}$. It is easy to verify that

$$\frac{1}{(1+(i-1)\gamma)^a - ((i-1)\gamma)^a} = O(i^{1-a})$$

given γ is a constant. Let t_i denote the item with maximum marginal gain in iteration i, i.e., $t_i \in \text{argmax}_{u \in V \setminus S_{i-1}} f(u|S_{i-1})$, and s_i denote the item selected by using the modular proxy $\hat{f}^{\text{mod}}$. Consider the following:

$$\begin{aligned}
\alpha_i &= \frac{f(t_i|S_{i-1})}{f(s_i|S_{i-1})} \leq \frac{f(t_i)}{f(s_i|S_{i-1})} \leq \frac{f(s_i)}{f(s_i|S_{i-1})} \\
&= \frac{\sum_{f \in \mathcal{F}} [c_f(s_i)]^a}{\sum_{f \in \mathcal{F}} [c_f(s_i) + c_f(S_{i-1})]^a - [c_f(S_{i-1})]^a} \\
&\leq \frac{\sum_{f \in \mathcal{F}} [c_f(s_i)]^a}{\sum_{f \in \mathcal{F}} [c_f(s_i) + (i-1)\rho_{\max}]^a - [(i-1)\rho_{\max}]^a}
\end{aligned}$$

Let's consider the following function:

$$h(x_1, \ldots, x_n) = \frac{\sum_{i=1}^{n} (x_i)^a}{\sum_{i=1}^{n} ((x_i + C)^a - C^a)} \tag{A.68}$$

where C is a constant. The function h is symmetric about all its variables $x_1, \ldots, x_n$. Notice that the function $h()$ is not convex or concave in its variables. However, we still want to maximize the function over the vector x within the range $[\rho_{\min}, \rho_{\max}]$ element-wisely. First, we easily see that h evaluated at $x_1 = x_2 = \cdots = x_n$ ranges in $(1, \infty)$.

To maximize the function h, we can simply maximize the function h over the subspace where $x_1 = \cdots = x_n$, since the maximum of h can be achieved in the subspace $x_1 = \cdots = x_n$. To maximize h, we can equivalently maximize the function $\frac{\sqrt{x}}{\sqrt{x+C} - \sqrt{C}}$ for $x \in [\rho_{\min}, \rho_{\max}]$, and it's easy to verify that the maximum is attained at $x = \rho_{\min}$. Moving back to upper bounding the greedy ratio at the ith iteration, we can get the following

$$\begin{aligned}
\alpha_i &\leq \frac{\rho_{\min}^a}{[\rho_{\min} + (i-1)\rho_{\max}]^a - [(i-1)\rho_{\max}]^a} \\
&= \frac{1}{[1 + (i-1)\gamma]^a - [(i-1)\gamma]^a}
\end{aligned}$$

□

A.15 Proof for Theorem 16

Theorem *If f_1 and f_2 are monotone submodular,* $\min\{f_1(A), f_2(V \setminus A)\}$ *is also submodular.*

Proof. To prove the Theorem, we show a more general result: Let f and h be submodular, and $f - h$ be either monotone increasing or decreasing, then $g(S) = \min\{f(S), h(S)\}$ is also submodular. The Theorem follows by this result, since $f(S) = f_1(S)$ and $h(S) = f_2(V \setminus S)$ are both submodular and $f(S) - h(S) = f_1(S) - f_2(V \setminus S)$ is monotone increasing.

In order to show g is submodular, we prove that g satisfies the following:

$$g(S) + g(T) \geq g(S \cap T) + g(S \cup T), \forall S, T \subseteq V \tag{A.69}$$

If g agrees with either f or h on both S and T, and since

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T) \tag{A.70}$$

$$h(S) + h(T) \geq h(S \cup T) + h(S \cap T), \tag{A.71}$$

$$\tag{A.72}$$

Eqn A.69 follows.

Otherwise, w.l.o.g. we consider $g(S) = f(S)$ and $g(T) = h(T)$. For the case where $f - h$ is monotone non-decreasing, consider the following:

$$g(S) + g(T) = f(S) + h(T) \tag{A.73}$$

$$\geq f(S \cup T) + f(S \cap T) - (f(T) - h(T)) \text{ // submodularity of } f \tag{A.74}$$

$$\geq f(S \cup T) + f(S \cap T) - (f(S \cup T) - h(S \cup T)) \text{ // monotonicity of } f - h \tag{A.75}$$

$$= f(S \cap T) + h(S \cup T) \tag{A.76}$$

$$\geq g(S \cap T) + g(S \cup T). \tag{A.77}$$

Similarly, for the case where $f - h$ is monotone non-increasing, consider the following:

$$g(S) + g(T) = f(S) + h(T) \tag{A.78}$$

$$\geq h(S \cap T) + h(S \cup T) + (f(S) - h(S)) \text{ // submodularity of } h \tag{A.79}$$

$$\geq h(S \cup T) + h(S \cap T) + (f(S \cup T) - h(S \cup T)) \text{ // monotonicity of } f - h \tag{A.80}$$

$$= h(S \cap T) + f(S \cup T) \tag{A.81}$$

$$\geq g(S \cap T) + g(S \cup T). \tag{A.82}$$

□

A.16 Proof for Theorem 17

Theorem *Under the homogeneous setting ($f_i = f$ for all i),* GreedMax *is guaranteed to find a partition $\hat{\pi}$ such that*

$$\min_{i=1,\dots,m} f(A_i^{\hat{\pi}}) \geq \frac{1}{m} \max_{\pi \in \Pi} \min_{i=1,\dots,m} f(A_i^{\pi}). \tag{A.83}$$

Proof. We prove that the guarantee of $1/m$, in fact, even holds for a streaming version of the greedy algorithm (StreamGreed, see Alg. 19). In particular, we show that StreamGreed provides a factor of $1/m$ for SFA under the homogeneous setting. Theorem 17 then follows since GreedMax can be seen as running StreamGreed with a specific order.

Algorithm 19: StreamGreed

```
Input: V = {v_1, v_2, ..., v_n}, f, m
Initialize: A_1 =, ..., = A_m = ∅, k = 1
while k ≤ n do
    i* ∈ argmin_j f(A_j)
    A_{i*} ← A_{i*} ∪ {v_k}
    k ← k + 1
```

To prove the guarantee for StreamGreed, we consider the resulting partitioning after an instance of StreamGreed: $\pi = (A_1^{\pi} \cup A_2^{\pi}, \dots, A_m^{\pi})$. For simplicity of notation, we write A_i^{π} as A_i for each i in the remaining proof. We refer OPT to the optimal solution, i.e.,

$OPT = \max_\pi \min_i f(A_i)$. W.l.o.g., we assume $f(A_1) = \min_i f(A_i)$. Let a_i be the last item to be chosen in block A_i for $i = 2, \ldots, m$.

Claim 1:

$$OPT \leq f(V \setminus \{a_2, \ldots, a_m\}) \tag{A.84}$$

To show this claim, consider the following: If we enlarge the singleton value of $a_i, i = 2, \ldots, m$, we obtain a new submodular function:

$$f'(A) = f(A) + \alpha \sum_{i=2}^{m} |A \cap a_i|, \tag{A.85}$$

where α is sufficiently large. Then running StreamGreed on f' with the same ordering of the incoming items leads to the same solution, since only the gain of the last added item for each block is changed.

Note that $f'(A) \geq f(A), \forall A \subseteq V$, we then have $\max_\pi \min_i f'(A_i^\pi) \geq OPT$. The optimal partitioning for f' can be easily obtained as $\pi' = (V \setminus \{a_2, \ldots, a_m\}, a_2, \ldots, a_m)$. Therefore, we have that

$$OPT \leq \max_\pi \min_i f'(A_i^\pi) = f'(V \setminus \{a_2, \ldots, a_m\}) = f(V \setminus \{a_2, \ldots, a_m\}). \tag{A.86}$$

Lastly, we have that $f(A_1) \geq f(A_i \setminus a_i)$ for any $i = 2, \ldots, m$ due to the procedure of StreamGreed. Therefore we have the following:

$$f(A_1) \geq \frac{1}{m}(f(A_1) + \sum_{i=2}^{m} f(A_i \setminus a_i)) \tag{A.87}$$

$$\geq \frac{1}{m} f(V \setminus \{a_2, \ldots, a_m\}) \text{ // submodularity of } f \tag{A.88}$$

$$\geq \frac{1}{m} OPT \text{ // Claim 1} \tag{A.89}$$

□

A.17 Proof for Theorem 18

Theorem *Given ϵ, α and any $0 < \delta < \alpha$,* GreedSat *finds a partition such that at least $\lceil m(\alpha - \delta) \rceil$ blocks receive utility at least $\frac{\delta}{1-\alpha+\delta}(\max_\pi \min_i f_i(A_i^\pi) - \epsilon)$.*

Proof. When GreedSat terminates, it identifies a $c_{\min}$ such that the returned solution $\hat{\pi}^{c_{\min}}$ satisfies $\bar{F}^{c_{\min}}(\hat{\pi}^{c_{\min}}) \geq \alpha c_{\min}$. Also it identifies a $c_{\max}$ such that the returned solution $\hat{\pi}^{c_{\max}}$ satisfies $\bar{F}^{c_{\max}}(\hat{\pi}^{c_{\max}}) < \alpha c_{\max}$. The gap between $c_{\max}$ and $c_{\min}$ is bounded by ϵ, i.e., $c_{\max} - c_{\min} \leq \epsilon$.

Next, we prove that there does not exist any partitioning π that satisfies $\min_i f_i(A_i^\pi) \geq c_{\max}$, i.e., $c_{\max} \geq \max_{\pi \in \Pi} \min_i f_i(A_i^\pi)$.

Suppose otherwise, i.e., $\exists \pi^* : \min_i f_i(A_i^{\pi^*}) = c_{\max} + \gamma$ with $\gamma \geq 0$. Let $c = c_{\max} + \gamma$, consider the intermediate objective $\bar{F}^c(\pi) = \frac{1}{m}\sum_{i=1}^m \min\{f_i(A_i^\pi), c\}$, we have that $\bar{F}^c(\pi^*) = c$. An instance of the algorithm for SWP on $\bar{F}^c$ is guaranteed to lead to a solution $\hat{\pi}^c$ such that $\bar{F}^c(\hat{\pi}^c) \geq \alpha c$. Since $c \geq c_{\max}$, it should follow that the returned solution $\hat{\pi}^{c_{\max}}$ for the value $c_{\max}$ also satisfies $\bar{F}^{c_{\max}}(\hat{\pi}) \geq \alpha c_{\max}$. However it contradicts with the termination criterion of GreedSat. Therefore, we prove that $c_{\max} \geq \max_{\pi \in \Pi} \min_i f_i(A_i^\pi)$, which indicates that $c_{\min} \geq c_{\max} - \epsilon \geq \max_{\pi \in \Pi} \min_i f_i(A_i^\pi) - \epsilon$.

Let $c^* = \frac{c_{\max} + c_{\min}}{2}$ and the partitioning returned by running for c^* be $\hat{\pi}$ (the final output partitioning from GreedSat). We have that $\bar{F}^{c^*}(\hat{\pi}) \geq \alpha c^*$, we are going to show that for any $0 < \delta < \alpha$, at least a $\lceil m(\alpha - \delta) \rceil$ blocks given by $\hat{\pi}$ receive utility larger or equal to $\frac{\delta}{1-\alpha+\delta} c^*$.

Just to restate the problem: we say that the i^{th} block is (α, δ)-good if $f_i(A_i^{\hat{\pi}}) \geq \frac{\delta}{1-\alpha+\delta} c^*$. Then the statement becomes: Given $0 < \delta < \alpha$, there is at least $m\lceil \alpha - \delta \rceil$ blocks that are (α, δ)-good.

To prove this statement, we assume, by contradiction, that there is strictly less than $\lceil m(\alpha - \delta) \rceil$ (α, δ)-good blocks. Denote the number of (α, δ)-good blocks as m_{good}. Then we have that $m_{\text{good}} \leq \lceil m(\alpha - \delta) \rceil - 1 < m(\alpha - \delta)$. Let $\theta = \frac{m_{\text{good}}}{m}$ be the fraction of (α, δ) good blocks, then we have that $0 \leq \theta < (\alpha - \delta) < 1$. The remaining fraction $(1 - \theta)$ of blocks are not good, i.e., they have valuation strictly less than $\frac{\delta}{1-\alpha+\delta} c^*$. Then, consider the following:

$$\bar{F}^{c^*}(\hat{\pi}) = \frac{1}{m} \sum_{i=1}^m \min\{f_i(A_i^{\hat{\pi}}), c^*\} \tag{A.90}$$

$$\overset{(a)}{<} \theta c^* + (1-\theta)\frac{\delta}{1-\alpha+\delta}c^* \tag{A.91}$$

$$= \frac{\delta}{1-\alpha+\delta}c^* + \frac{1-\alpha}{1-\alpha+\delta}\theta c^* \tag{A.92}$$

$$\overset{(b)}{<} \frac{\delta}{1-\alpha+\delta}c^* + \frac{1-\alpha}{1-\alpha+\delta}(\alpha-\delta)c^* \tag{A.93}$$

$$= \alpha c^* \tag{A.94}$$

Inequality (a) follows since good blocks are upper bounded by c^*, and not good blocks have values upper bounded by $\frac{\delta}{1-\alpha+\delta}c^*$. Inequality (b) follows by the assumption on θ. This therefore contradicts the assumption that $\bar{F}^{c^*}(\hat{\pi}) \geq \alpha c^*$, hence the statement is true.

This statement can also be proved using a different strategy. Let $f_i^{c^*} = \min\{f_i(A_i^{\hat{\pi}}), c^*\}$ and $f_i = f_i(A_i^{\hat{\pi}})$ for all i. For any $0 \leq \beta \leq 1$ the following holds:

$$\alpha c^* \leq \bar{F}^{c^*}(\hat{\pi}) = \frac{1}{m}\sum_i f_i^{c^*} \leq \frac{1}{m}f_i = \frac{1}{m}\sum_{i:f_i<\beta c^*} f_i + \frac{1}{m}\sum_{i:f_i\geq\beta c^*} f_i < \frac{1}{m}m_{\text{bad}}\beta c^* + \frac{1}{m}m_{\text{good}}c^* \tag{A.95}$$

where $m = m_{\text{bad}} + m_{\text{good}}$ and m_{good} are the number that are β-good (i.e., i is β-good if $f_i \geq \beta c^*$). The goal is to place a lower bound on m_{good}. From the above

$$\alpha < (1 - \frac{m_{\text{good}}}{m})\beta + \frac{m_{\text{good}}}{m} \tag{A.96}$$

which means

$$m_{\text{good}} \geq \lceil \frac{\alpha-\beta}{1-\beta}m \rceil \tag{A.97}$$

Let $\beta = \frac{\delta}{1-\alpha+\delta}$, the statement immediately follows.

Note $c^* = \frac{c_{\min}+c_{\max}}{2} \geq c_{\max} - \epsilon \geq \max_{\pi\in\Pi}\min_i f_i(A_i^{\pi}) - \epsilon$. Combining pieces together, we have shown that at least $\lceil m(\alpha-\delta)\rceil$ blocks given by $\hat{\pi}$ receive utility larger or equal to $\frac{\delta}{1-\alpha+\delta}(\max_{\pi\in\Pi}\min_i f_i(A_i^{\pi}) - \epsilon)$. □

A.18 Proof for Theorem 19

Theorem MMax *achieves a worst-case guarantee of* $O(\min_i \frac{1+(|A_i^{\hat{\pi}}|-1)(1-\kappa_{f_i}(A_i^{\hat{\pi}}))}{|A_i^{\hat{\pi}}|\sqrt{m}\log^3 m})$*, where* $\hat{\pi} = (A_1^{\hat{\pi}}, \cdots, A_m^{\hat{\pi}})$ *is the partition obtained by the algorithm, and* $\kappa_f(A) = 1 - \min_{v\in V}\frac{f(v|A\backslash v)}{f(v)} \in [0,1]$ *is the curvature of a submodular function* f *at* $A \subseteq V$*.*

Proof. We assume the approximation factor of the algorithm for solving the modular version of Problem 1 is $\alpha = O(\frac{1}{\sqrt{m}\log^3 m})$ [Asadpour and Saberi, 2010]. For notation simplicity, we write $\hat{\pi} = (\hat{A}_1, \ldots, \hat{A}_m)$ as the resulting partition after the first iteration of MMax, and $\pi^* = (A_1^*, \ldots, A_m^*)$ as its optimal solution. Note that first iteration suffices to yield the performance guarantee, and the subsequent iterations are designed so as to improve the empirical performance. Since the proxy function for each function f_i used for the first iteration are the simple modular upper bound with the form: $h_i(X) = \sum_{j \in X} f_i(j)$.

Given the curvature of each submodular function f_i, we can tightly bound a submodular function f_i in the following form [Iyer et al., 2013a]:

$$f_i(X) \leq h_i(X) \leq \frac{|X|}{1 + (|X| - 1)(1 - \kappa_{f_i}(X))} f_i(X), \forall X \subseteq V \tag{A.98}$$

Consider the following:

$$\min_i f_i(\hat{A}_i) \geq \min_i \frac{1}{\frac{|\hat{A}_i|}{1+(|\hat{A}_i|-1)(1-\kappa_{f_i}(\hat{A}_i))}} h_i(\hat{A}_i) \tag{A.99}$$

$$\geq \min_i \frac{1}{\frac{|\hat{A}_i|}{1+(|\hat{A}_i|-1)(1-\kappa_{f_i}(\hat{A}_i))}} \min_i h_i(\hat{A}_i) \tag{A.100}$$

$$\geq \alpha \min_i \frac{1 + (|\hat{A}_i| - 1)(1 - \kappa_{f_i}(\hat{A}_i))}{|\hat{A}_i|} \min_i h_i(A_i^*) \tag{A.101}$$

$$\geq \alpha \min_i \frac{1 + (|\hat{A}_i| - 1)(1 - \kappa_{f_i}(\hat{A}_i))}{|\hat{A}_i|} \min_i f_i(A_i^*) \tag{A.102}$$

$$= O(\min_i \frac{1 + (|\hat{A}_i| - 1)(1 - \kappa_{f_i}(\hat{A}_i))}{|\hat{A}_i|\sqrt{m}\log^3 m}) \min_i f_i(A_i^*). \tag{A.103}$$

□

A.19 Proof for Theorem 20

Theorem *Suppose there exists an algorithm for solving the modular version of SFA with an approximation factor $\alpha \leq 1$, we have that*

$$\min_i f_i(A_i^{\pi_t}) \geq \alpha \min_i f_i(A_i^{\pi_{t-1}}). \tag{A.104}$$

Proof. Consider the following:

$$\min_i f_i(A_i^{\pi_{t-1}}) = \min_i h_i(A_i^{\pi_{t-1}}) // \text{ tightness of modular lower bound.} \tag{A.105}$$

$$\leq \alpha \min_i h_i(A_i^{\pi_t}) // \text{ approximation factor of the modular SFA.} \tag{A.106}$$

$$\leq \alpha h_j(A_j^{\pi_t}) // j \in \operatorname*{argmin}_i f_i(A_i^{\pi_t}) \tag{A.107}$$

$$\leq \alpha f_j(A_j^{\pi_t}) // h_j(A_j^{\pi_{t-1}}) \text{ upper bounds } f_j \text{ everywhere.} \tag{A.108}$$

$$= \alpha \min_i f_i(A_i^{\pi_t}) \tag{A.109}$$

□

A.20 Proof for Theorem 21

Theorem *For any $\epsilon > 0$, SLB cannot be approximated to a factor of $(1-\epsilon)m$ for any $m = o(\sqrt{n/\log n})$ with polynomial number of queries even under the homogeneous setting.*

Proof. We use the same proof techniques as in [Svitkina and Fleischer, 2008]. Consider two submodular functions:

$$f_1(S) = \min\{|S|, \alpha\}; \tag{A.110}$$

$$f_2(S) = \min\{\sum_{i=1}^{m} \min\{\beta, |S \cap V_i|\}, \alpha\}; \tag{A.111}$$

where $\{V_i\}_{i=1}^m$ is a uniformly random partitioning of V into m blocks, $\alpha = \frac{n}{m}$ and $\beta = \frac{n}{m^2(1-\epsilon)}$. To be more precise about the uniformly random partitioning, we assign each item into any one of the m blocks with probability $1/m$. It can be easily verified that $OPT_1 = \min_{\pi\in\Pi} \max_i f_1(A_i^\pi) = n/m$ and $OPT_2 = \min_{\pi\in\Pi} \max_i f_2(A_i^\pi) = \frac{n}{m^2(1-\epsilon)}$. The gap is then $\frac{OPT_1}{OPT_2} = m(1-\epsilon)$.

Next, we show that f_1 and f_2 cannot be distinguished with $n^{\omega(1)}$ number of queries.

Since $f_1(S) \geq f_2(S)$ holds for any S, this is equivalent as showing $P\{f_1(S) > f_2(S)\} < n^{-\omega(1)}$.

As shown in [Svitkina and Fleischer, 2008], $P\{f_1(S) > f_2(S)\}$ is maximized when $|S| = \alpha$. It suffices to consider only the case of $|S| = \alpha$ as follows:

$$P\{f_1(S) > f_2(S) : |S| = \alpha\} = P\{\sum_{i=1}^{m} \min\{\beta, |S \cap V_i|\} < \alpha : |S| = \alpha\} \tag{A.112}$$

The necessary condition for $\sum_{i=1}^{m} \min\{\beta, |S \cap V_i|\} < \alpha$ is that $|S \cap V_i| > \beta$ is satisfied for some i. Using the Chernoff bound, we have that for any i, it holds $P\{|S \cap V_i| > \beta\} \leq e^{-\frac{\epsilon^2 n}{3m^2}} = n^{-\omega(1)}$ when $m = o(\sqrt{n/\log n})$. Using the union bound, it holds that the probability for any one block V_i such that $|S \cap V_i| > \beta$ is also upper bounded by $n^{-\omega(1)}$. Combining all pieces together, we have the following:

$$P\{f_1(S) > f_2(S)\} \leq n^{-\omega(1)}. \tag{A.113}$$

Finally, we come to prove the Theorem. Suppose the goal is to solve an instance of SLB with f_2. Since f_1 and f_2 are hard to distinguish with polynomial number of function queries, any polynomial time algorithm for solving f_2 is equivalent to solving for f_1. However, the optimal solution for f_1 is $\alpha = \frac{n}{m}$, whereas the optimal solution for f_2 is $\beta = \frac{n}{m^2(1-\epsilon)}$. Therefore, no polynomial time algorithm can find a solution with a factor $m(1-\epsilon)$ for SLB in this case. □

A.21 Proof for Theorem 22

Theorem LovászRound *is guaranteed to find a partition* $\hat{\pi} \in \Pi$ *such that* $\max_i f_i(A_i^{\hat{\pi}}) \leq m \min_{\pi \in \Pi} \max_i f_i(A_i^{\pi})$.

Proof. It suffices to bound the performance loss at the step of rounding the fractional solution $\{x_i^*\}_{i=1}^m$, or equivalently, the following:

$$\max_i \tilde{f}_i(x_i^*) \geq \frac{1}{m} \max_i f_i(A_i), \tag{A.114}$$

where $\{A_i\}_{i=1}^m$ is the resulting partitioning after the rounding. To show Eqn A.114, it suffices to show that $\tilde{f}_i(x_i^*) \geq \frac{1}{m} f_i(A_i)$ for all $i = 1, \ldots, m$. Next, consider the following:

$$f_i(A_i) = \tilde{f}_i(1_{A_i}) = m\tilde{f}_i(\frac{1}{m}1_{A_i}) // \text{ positive homogeneity of Lovász extension} \tag{A.115}$$

For any item $v_j \in A_i$, we have $x_i^*(j) \geq \frac{1}{m}$, since $\sum_{i=1}^m x_i^*(j) \geq 1$ and $x_i^*(j) = \max_{i'} x_{i'}(j)$. Therefore, we have $\frac{1}{m} 1_{A_i} \leq x_i^*$. Since f_i is monotone, its extension $\tilde{f}_i$ is also monotone. As a result, $f_i(A_i) = m\tilde{f}_i(\frac{1}{m}1_{A_i}) \leq m\tilde{f}_i(x_i^*)$.

□

A.22 Proof for Theorem 23

Theorem MMin *achieves a worst-case guarantee of* $(2\max_i \frac{|A_i^{\pi^*}|}{1+(|A_i^{\pi^*}|-1)(1-\kappa_{f_i}(A_i^{\pi^*}))})$, *where* $\pi^* = (A_1^{\pi^*}, \cdots, A_m^{\pi^*})$ *denotes the optimal partition.*

Proof. Let $\alpha = 2$ be the approximation factor of the algorithm for solving the modular version of Problem 2 [Lenstra et al., 1990]. For notation simplicity, we write $\hat{\pi} = (\hat{A}_1, \ldots, \hat{A}_m)$ as the resulting partition after the first iteration of MMin, and $\pi^* = (A_1^*, \ldots, A_m^*)$ as its optimal solution. Again the first iteration suffices to yield the performance guarantee, and the subsequent iterations are designed so as to improve the empirical performance. Since the supergradients for each function f_i used for the first iteration are the simple modular upper bound with the form: $h_i(X) = \sum_{j \in X} f_i(j)$, we can again tightly bound a submodular function f_i in the following form:

$$f_i(X) \leq h_i(X) \leq \frac{|X|}{1 + (|X| - 1)(1 - \kappa_{f_i}(X))} f_i(X), \forall X \subseteq V \tag{A.116}$$

Consider the following:

$$\max_i f_i(\hat{A}_i) \leq \max_i h_i(\hat{A}_i) \tag{A.117}$$

$$\leq \alpha \max_i h_i(A_i^*) \tag{A.118}$$

$$\leq \alpha \max_i \frac{|A_i^*|}{1 + (|A_i^*| - 1)(1 - \kappa_{f_i}(A_i^*))} f_i(A_i^*) \tag{A.119}$$

$$\leq \alpha \max_i \frac{|A_i^*|}{1 + (|A_i^*| - 1)(1 - \kappa_{f_i}(A_i^*))} \max_i f_i(A_i^*) \tag{A.120}$$

□

A.23 Proof for Theorem 24

Theorem *Suppose there exists an algorithm for solving the modular version of SLB with an approximation factor $\alpha \geq 1$, we have for each iteration t that*

$$\max_i f_i(A_i^{\pi_t}) \leq \alpha \max_i f_i(A_i^{\pi_{t-1}}). \tag{A.121}$$

Proof. The proof is symmetric to the one for Theorem 20. □

A.24 Proof for Theorem 25

We prove separately for Problem 7.1 and Problem 7.2.

Theorem *Given an instance of Problem 7.1 with $0 < \lambda < 1$,* CombSfaSwp *provides an approximation guarantee of* $\max\{\min\{\alpha, \frac{1}{m}\}, \frac{\beta\alpha}{\lambda\beta+\alpha}, \lambda\beta\}$ *in the* homogeneous *case, and a factor of* $\max\{\frac{\beta\alpha}{\lambda\beta+\alpha}, \lambda\beta\}$ *in the* heterogeneous *case, where α and β are the approximation factors of* AlgWC *and* AlgAC *for SFA and SWP respectively.*

Proof. We first prove the result for heterogeneous setting. For notation simplicity we write the worst-case objective as $F_1(\pi) = \min_{i=1,\ldots,m} f(A_i^\pi)$ and the average-case objective as $F_2(\pi) = \frac{1}{m}\sum_{i=1,\ldots,m} f(A_i^\pi)$.

Suppose AlgWC outputs a partition $\hat{\pi}_1$ and AlgAC outputs a partition $\hat{\pi}_2$. Let $\pi^* \in \arg\max_{\pi\in\Pi} \bar{\lambda}F_1(\pi) + \lambda F_2(\pi)$ be the optimal partition.

We use the following facts:

Fact1

$$F_1(\hat{\pi}_1) \geq \alpha F_1(\pi) \tag{A.122}$$

Fact2

$$F_2(\pi_2) \geq \beta F_2(\pi) \tag{A.123}$$

Fact3

$$F_1(\pi) \leq F_2(\pi) \tag{A.124}$$

Then we have that

$$\begin{aligned}
\bar{\lambda}F_1(\hat{\pi}_2) + \lambda F_2(\hat{\pi}_2) &\geq \lambda F_2(\hat{\pi}_2) && \text{(A.125)}\\
&\geq \lambda\beta F_2(\pi^*) && \text{(A.126)}\\
&\geq \lambda\beta \left[\bar{\lambda}F_1(\pi^*) + \lambda F_2(\pi^*)\right] && \text{(A.127)}
\end{aligned}$$

and

$$\begin{aligned}
\bar{\lambda}F_1(\hat{\pi}_1) + \lambda F_2(\hat{\pi}_1) &\geq \mu\left[\bar{\lambda}F_1(\hat{\pi}_1) + \lambda F_2(\hat{\pi}_1)\right] + (1-\mu)\left[\bar{\lambda}F_1(\hat{\pi}_1) + \lambda F_2(\hat{\pi}_1)\right] && \text{(A.128)}\\
&\geq \mu\left[\bar{\lambda}\alpha F_1(\pi^*) + \lambda\alpha F_1(\pi^*)\right] + (1-\mu)\left[0 + \lambda\beta F_2(\pi^*)\right] && \text{(A.129)}\\
&\geq \frac{\mu\alpha}{\bar{\lambda}}\bar{\lambda}F_1(\pi^*) + (1-\mu)\beta\lambda F_2(\pi^*) && \text{(A.130)}\\
&\geq \min\{\frac{\mu\alpha}{\bar{\lambda}}, (1-\mu)\beta\}\left[\bar{\lambda}F_1(\pi^*) + \lambda F_2(\pi^*)\right] && \text{(A.131)}
\end{aligned}$$

$\min\{\frac{\mu\alpha}{\bar{\lambda}}, (1-\mu)\beta\}$ is a function over $0 \leq \mu \leq 1$ and $\mu^* \in \arg\max_\mu \min\{\frac{\mu\alpha}{\bar{\lambda}}, (1-\mu)\beta\}$. It is easy to show

$$\begin{aligned}
\mu^* &= \frac{\bar{\lambda}\beta}{\bar{\lambda}\beta + \alpha} && \text{(A.132)}\\
\max_\mu \min\{\frac{\mu\alpha}{\bar{\lambda}}, (1-\mu)\beta\} &= \frac{\beta\alpha}{\bar{\lambda}\beta + \alpha} && \text{(A.133)}
\end{aligned}$$

$$\bar{\lambda}F_1(\hat{\pi}_1) + \lambda F_2(\hat{\pi}_1) \geq \frac{\beta\alpha}{\bar{\lambda}\beta + \alpha}\left[\bar{\lambda}F_1(\pi^*) + \lambda F_2(\pi^*)\right] \tag{A.134}$$

Taking the max over the two bounds leads to

$$\max\{\bar{\lambda}F_1(\hat{\pi}_1) + \lambda F_2(\hat{\pi}_1), \bar{\lambda}F_1(\hat{\pi}_2) + \lambda F_2(\hat{\pi}_2)\} \geq \max\{\frac{\beta\alpha}{\bar{\lambda}\beta + \alpha}, \lambda\beta\} \max_{\pi\in\Pi} \bar{\lambda}F_1(\pi) + \lambda F_2(\pi) \tag{A.135}$$

Next we are going to show the result for the homogeneous setting. We have the following facts that hold for arbitrary partition π:

$$F_1(\hat{\pi}_1) \geq \alpha F_1(\pi),\ F_2(\hat{\pi}_2) \geq \beta F_2(\pi) \tag{A.136}$$

$$F_1(\pi) \leq F_2(\pi),\ F_2(\pi_1) \geq \frac{1}{m} F_2(\pi) \tag{A.137}$$

Consider the following:

$$\bar{\lambda} F_1(\hat{\pi}_1) + \lambda F_2(\hat{\pi}_1) \geq \alpha \bar{\lambda} F_1(\pi^*) + \frac{\lambda}{m} F_2(\pi^*) \tag{A.138}$$

$$\geq \min\{\alpha, \frac{1}{m}\} \left[\bar{\lambda} F_1(\pi^*) + \lambda F_2(\pi^*)\right] \tag{A.139}$$

and

$$\bar{\lambda} F_1(\hat{\pi}_2) + \lambda F_2(\hat{\pi}_2) \geq \lambda F_2(\hat{\pi}_2) \tag{A.140}$$

$$\geq \lambda\beta F_2(\pi^*) \tag{A.141}$$

$$\geq \lambda\beta \left[\bar{\lambda} F_1(\pi^*) + \lambda F_2(\pi^*)\right] \tag{A.142}$$

Taking the max over the two bounds and the result shown in Eqn A.134 gives the following:

$$\max\{\bar{\lambda} F_1(\hat{\pi}_1) + \lambda F_2(\hat{\pi}_1), \bar{\lambda} F_1(\hat{\pi}_2) + \lambda F_2(\hat{\pi}_2)\} \geq \max\{\min\{\alpha, \frac{1}{m}\}, \frac{\beta\alpha}{\bar{\lambda}\beta + \alpha}, \lambda\beta\} \max_{\pi \in \Pi} \bar{\lambda} F_1(\pi) + \lambda F_2(\pi). \tag{A.143}$$

□

Theorem CombSlbSmp *provides an approximation guarantee of* $\min\{m, \frac{m\alpha}{m\bar{\lambda}+\lambda}, \beta(m\bar{\lambda}+\lambda)\}$ *in the* homogeneous *case, and a factor of* $\min\{\frac{m\alpha}{m\bar{\lambda}+\lambda}, \beta(m\bar{\lambda}+\lambda)\}$ *in the* heterogeneous *case, for Problem 7.2 with* $0 \leq \lambda \leq 1$.

Proof. Let $\hat{\pi}_1$ be the solution of AlgWC and $\hat{\pi}_2$ be the solution of AlgAC. Let $\pi^* \in \arg\min_{\pi \in \Pi} \bar{\lambda} F_1(\pi) + \lambda F_2(\pi)$ be the optimal partition. The following facts hold for all $\pi \in \Pi$:

Fact1

$$F_1(\hat{\pi}_1) \leq \alpha F_1(\pi) \tag{A.144}$$

Fact2

$$F_2(\hat{\pi}_2) \leq \beta F_2(\pi) \tag{A.145}$$

Fact3

$$F_2(\pi) \leq F_1(\pi) \leq mF_2(\pi) \tag{A.146}$$

Then we have the following:

$$\begin{aligned}
\bar{\lambda}F_1(\hat{\pi}_1) + \lambda F_2(\hat{\pi}_1) &\leq F_1(\hat{\pi}_1) && \text{(A.147)}\\
&\leq \alpha F_1(\pi^*) && \text{(A.148)}\\
&\leq \frac{\alpha}{\bar{\lambda} + \frac{\lambda}{m}} \left[\bar{\lambda}F_1(\pi^*) + \frac{\lambda}{m}F_1(\pi^*)\right] && \text{(A.149)}\\
&\leq \frac{m\alpha}{m\bar{\lambda} + \lambda} \left[\bar{\lambda}F_1(\pi^*) + \lambda F_2(\pi^*)\right] && \text{(A.150)}
\end{aligned}$$

and

$$\begin{aligned}
\bar{\lambda}F_1(\hat{\pi}_2) + \lambda F_2(\hat{\pi}_2) &\leq \beta m\bar{\lambda}F_2(\pi^*) + \beta\lambda F_2(\pi^*) && \text{(A.151)}\\
&\leq (m\bar{\lambda} + \lambda)\beta F_2(\pi^*) && \text{(A.152)}\\
&\leq (m\bar{\lambda} + \lambda)\beta \left[\bar{\lambda}F_1(\pi^*) + \lambda F_2(\pi^*)\right] && \text{(A.153)}\\
& && \text{(A.154)}
\end{aligned}$$

Taking the minimum over the two leads to the following:

$$\min\{\bar{\lambda}F_1(\hat{\pi}_1) + \lambda F_2(\hat{\pi}_1), \bar{\lambda}F_1(\hat{\pi}_2) + \lambda F_2(\hat{\pi}_2)\} \leq \min\{\frac{m\alpha}{m\bar{\lambda} + \lambda}, \beta(m\bar{\lambda} + \lambda)\} \min_{\pi \in \Pi} \bar{\lambda}F_1(\pi) + \lambda F_2(\pi) \tag{A.155}$$

Equation A.155 gives us a bound for both the homogeneous setting and the heterogeneous settings.

Furthermore, in the homogeneous setting, for arbitrary partition π, we have

$$\bar{\lambda}F_1(\pi) + \lambda F_2(\pi) \leq m \min_{\pi \in \Pi} \bar{\lambda}F_1(\pi) + \lambda F_2(\pi) \tag{A.156}$$

and we can tighten the bound for the homogeneous setting as follows:

$$\min\{\bar{\lambda}F_1(\hat{\pi}_1)+\lambda F_2(\hat{\pi}_1), \bar{\lambda}F_1(\hat{\pi}_2)+\lambda F_2(\hat{\pi}_2)\} \leq \min\{m, \frac{m\alpha}{m\bar{\lambda}+\lambda}, \beta(m\bar{\lambda}+\lambda)\} \max_{\pi\in\Pi} \bar{\lambda}F_1(\pi)+\lambda F_2(\pi) \tag{A.157}$$

□

A.25 Proof for Theorem 26

Theorem *Given ϵ, α, and, $0 \leq \lambda \leq 1$,* GeneralGreedSat *finds a partition $\hat{\pi}$ that satisfies the following:*

$$\bar{\lambda} \min_i f_i(A_i^{\hat{\pi}}) + \lambda \frac{1}{m} \sum_{i=1}^{m} f_i(A_i^{\hat{\pi}}) \geq \lambda\alpha(OPT-\epsilon), \tag{A.158}$$

where $OPT = \max_{\pi\in\Pi} \bar{\lambda} \min_i f_i(A_i^{\pi}) + \lambda\frac{1}{m}\sum_{i=1}^{m} f_i(A_i^{\pi})$.

Moreover, let $F_{\lambda,i}(\pi) = \bar{\lambda}f_i(A_i^{\pi}) + \lambda\frac{1}{m}\sum_{j=1}^{m} f_j(A_j^{\pi})$. Given any $0 < \delta < \alpha$, there is a set $I \subseteq \{1,\ldots,m\}$ such that $|I| \geq \lceil m(\alpha-\delta)\rceil$ and

$$F_{i,\lambda}(\hat{\pi}) \geq \max\{\frac{\delta}{1-\alpha+\delta}, \lambda\alpha\}(OPT-\epsilon), \forall i \in I. \tag{A.159}$$

Proof. Denote intermediate objective $\bar{F}^c(\pi) = \frac{1}{m}\sum_{i=1}^{m} \min\{\bar{\lambda}f_i(A_i^{\pi}) + \lambda\frac{1}{m}\sum_{j=1}^{m} f_j(A_j^{\pi}), c\}$. Also we define the overall objective as $F(\pi) = \bar{\lambda}\min_i f_i(A_i^{\pi}) + \lambda\frac{1}{m}\sum_{i=1}^{m} f_i(A_i^{\pi})$. When the algorithm terminates, it identifies a $c_{\min}$ such that the returned solution $\hat{\pi}^{c_{\min}}$ satisfies $\bar{F}^{c_{\min}}(\hat{\pi}^{c_{\min}}) \geq \alpha c_{\min}$. Also it identifies a $c_{\max}$ such that the returned solution $\hat{\pi}^{c_{\max}}$ satisfies $\bar{F}^{c_{\max}}(\hat{\pi}^{c_{\max}}) < \alpha c_{\max}$. The gap between $c_{\max}$ and $c_{\min}$ is bounded by ϵ, i.e., $c_{\max} - c_{\min} \leq \epsilon$.

Next, we prove that there does not exist any partitioning π that satisfies $F(\pi) \geq c_{\max}$, i.e., $c_{\max} \geq OPT$.

Suppose otherwise, i.e., $\exists\pi^* : F(\pi^*) = c_{\max} + \gamma$ with $\gamma \geq 0$. Let $c = c_{\max} + \gamma$, consider the intermediate objective $\bar{F}^c(\pi)$, we have that $\bar{F}^c(\pi^*) = c$. An instance of the algorithm for SWP on $\bar{F}^c$ is guaranteed to lead to a solution $\hat{\pi}^c$ such that $\bar{F}^c(\hat{\pi}^c) \geq \alpha c$. Since $c \geq c_{\max}$, it should

follow that the returned solution $\hat{\pi}^{c_{\max}}$ for the value $c_{\max}$ also satisfies $\bar{F}^{c_{\max}}(\hat{\pi}) \geq \alpha c_{\max}$. However it contradicts with the termination criterion of GreedSat. Therefore, we prove that $c_{\max} \geq OPT$, which indicates that $c_{\min} \geq c_{\max} - \epsilon \geq OPT - \epsilon$.

Let $c^* = \frac{c_{\max}+c_{\min}}{2}$ and the partitioning returned by running for c^* be $\hat{\pi}$ (the final output partitioning from the algorithm). We have that $\bar{F}^{c^*}(\hat{\pi}) \geq \alpha c^*$.

Next we are ready to prove the Theorem: $F(\hat{\pi}) \geq \lambda\alpha$. For simplicity of notation, we rewrite $y_i = \bar{\lambda} f_i(A_i^{\hat{\pi}}) + \lambda \frac{1}{m} \sum_{j=1}^m f_j(A_j^{\hat{\pi}})$ and $x_i = \min\{\bar{\lambda} f_i(A_i^{\hat{\pi}}) + \lambda \frac{1}{m} \sum_{j=1}^m f_j(A_j^{\hat{\pi}}), c^*\} = \min\{y_i, c^*\}$ for each i. Furthermore, we denote the sample mean as $\bar{x} = \frac{1}{m}\sum_{i=1}^m x_i$ and $\bar{y} = \frac{1}{m}\sum_{i=1}^m y_i$. Then, we have $F(\hat{\pi}) = \min_i y_i$ and $\bar{F}^{c^*}(\hat{\pi}) = \bar{x}$. We list the following facts to facilitate the analysis:

1. $0 \leq x_i \leq c^*$ holds for all i;

2. $y_i \geq \lambda \bar{y}$ holds for all i;

3. $x_i \geq \lambda \bar{x}$ holds for all i;

4. $\bar{x} \geq \alpha c^*$;

5. $x_i = \min\{y_i, c^*\}, \forall i$.

The second fact follows since

$$\bar{y} = \frac{1}{m}\sum_{i=1}^m y_i \tag{A.160}$$

$$= \frac{1}{m}\sum_{i=1}^m \{\bar{\lambda} f_i(A_i^{\hat{\pi}}) + \lambda \frac{1}{m}\sum_{j=1}^m f_j(A_j^{\hat{\pi}})\} \tag{A.161}$$

$$= \frac{1}{m}\sum_{j=1}^m f_j(A_j^{\hat{\pi}}) \leq \frac{y_i}{\lambda} \tag{A.162}$$

Given the second fact, we can prove the third fact as follows. Let $i^* \in \operatorname{argmin}_i y_i$. By definition $x_i = \min\{y_i, c^*\}$, then $i^* \in \operatorname{argmin}_i x_i$. We consider the two cases:

(1) $y_{i^*} \leq c^*$: In this case, we have that $x_{i^*} = y_{i^*}$. Since $x_i \leq y_i, \forall i$, it holds that $\bar{x} \leq \bar{y}$. The third fact follows as $x_{i^*} = y_{i^*} \geq \lambda \bar{y} \geq \lambda \bar{x}$.

(2) $y_{i^*} > c^*$: In this case, $y_i \geq c^*$ holds for all i. As a result, we have $x_i = c^*, \forall i$. Therefore, $x_i = \bar{x} = c^* \geq \lambda c^*$.

Combining fact 3 and 4, it follows for each i:

$$\bar{\lambda} f_i(A_i^{\hat{\pi}}) + \lambda \frac{1}{m} \sum_{j=1}^{m} f_j(A_j^{\hat{\pi}}) = y_i \geq x_i \geq \lambda \bar{x} \geq \alpha \lambda c^* \geq \alpha \lambda (OPT - \epsilon). \tag{A.163}$$

The first part of the Theorem is then proved.

The second part of the Theorem simply follows from the proof in Theorem 18 and Eqn A.163. □

A.26 Proof for Theorem 27

Theorem *Define* $F^{\lambda}(\pi) = \bar{\lambda} \max_i f_i(A_i^{\pi}) + \lambda \frac{1}{m} \sum_{i=1}^{m} f_i(A_i^{\pi})$ *for any* $0 \leq \lambda \leq 1$. GENERALLOVÁSZ ROUND *is guaranteed to find a partition* $\hat{\pi} \in \Pi$ *such that*

$$F^{\lambda}(\hat{\pi}) \leq m \min_{\pi \in \Pi} F^{\lambda}(\pi) \tag{A.164}$$

Proof. We essentially use the same proof technique in Theorem 22 to show this result. After solving for the continuous solution $\{x_i^* \in \mathbb{R}^n\}_{i=1}^m$, the rounding step simply chooses for each $j = 1, \ldots, n$, assigns the item to the block $i^* \in \text{argmax}_{i=1,\ldots,m} x_i^*(j)$. We denote the resulting partitioning as $\hat{\pi} = \{A_i^{\hat{\pi}}\}_{i=1}^m$.

It suffices to bound the performance loss at the step of rounding the fractional solution $\{x_i^*\}_{i=1}^m$, or equivalently, the following:

$$\tilde{f}_i(x_i^*) \geq \frac{1}{m} f_i(A_i^{\hat{\pi}}), \tag{A.165}$$

Given Eqn A.165, the Theorem follows since

$$F^{\lambda}(\pi^*) \geq \bar{\lambda} \max_i \tilde{f}_i(x_i^*) + \lambda \frac{1}{m} \sum_{j=1}^{m} \tilde{f}_j(x_j^*) \tag{A.166}$$

$$\geq \frac{1}{m}[\bar{\lambda} \max_i f_i(A_i^{\hat{\pi}}) + \frac{\lambda}{m} \sum_{j=1}^{m} f_j(A_j^{\hat{\pi}})] \tag{A.167}$$

$$\geq \frac{1}{m} F^{\lambda}(\hat{\pi}). \tag{A.168}$$

To prove Eqn A.114, consider the following:

$$f_i(A_i^{\hat{\pi}}) = \tilde{f}_i(1_{A_i^{\hat{\pi}}}) = m\tilde{f}_i(\frac{1}{m}1_{A_i^{\hat{\pi}}}) // \text{ positive homogeneity of Lovász extension} \tag{A.169}$$

For any item $v_j \in A_i^{\hat{\pi}}$, we have $x_i^*(j) \geq \frac{1}{m}$, since $\sum_{i=1}^{m} x_i^*(j) \geq 1$ and $x_i^*(j) = \max_{i'} x_{i'}(j)$. Therefore, we have $\frac{1}{m}1_{A_i} \leq x_i^*$. Since f_i is monotone, its extension $\tilde{f}_i$ is also monotone. As a result, $f_i(A_i) = m\tilde{f}_i(\frac{1}{m}1_{A_i}) \leq m\tilde{f}_i(x_i^*)$. □