Active Learning and Submodular Functions

Andrew Guillory

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

Doctor of Philosophy

University of Washington

2012

Reading Committee:

Jeff Bilmes, Chair

Anna Karlin

Marina Meila

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Active Learning and Submodular Functions

Andrew Guillory

Chair of the Supervisory Committee:
Professor Jeff Bilmes
Electrical Engineering

Active learning is a machine learning setting where the learning algorithm decides what data is labeled. Submodular functions are a class of set functions for which many optimization problems have efficient exact or approximate algorithms. We examine their connections.

- We propose a new class of interactive submodular optimization problems which connect and generalize submodular optimization and active learning over a finite query set. We derive greedy algorithms with approximately optimal worst-case cost. These analyses apply to exact learning, approximate learning, learning in the presence of adversarial noise, and applications that mix learning and covering.
- We consider active learning in a batch, transductive setting where the learning algorithm selects a set of examples to be labeled at once. In this setting we derive new error bounds which use symmetric submodular functions for regularization, and we give algorithms which approximately minimize these bounds.
- We consider a repeated active learning setting where the learning algorithm solves a sequence of related learning problems. We propose an approach to this problem based on a new online prediction version of submodular set cover.

A common theme in these results is the use of tools from submodular optimization to extend the breadth and depth of learning theory with an emphasis on non-stochastic settings.

TABLE OF CONTENTS

Page

List of Figures iii

List of Tables iv

Chapter 1: Introduction 1
1.1 Motivation 1
1.2 Background: Active Learning 2
1.3 Background: Submodular Functions 7
1.4 A Simple Approach 15
1.5 Our Contributions 16

Chapter 2: Interactive Submodular Optimization 18
2.1 Problems 19
2.2 Worst Case Cost and Value 20
2.3 Greedy Algorithm 22
2.4 Analysis 24
2.5 Explicit Coverage Threshold 30
2.6 Restricted Adversary 31
2.7 Connection to Exact Active Learning 35
2.8 Hardness 38
2.9 Adaptivity Gap 38

Chapter 3: Multiple Objectives 40
3.1 Problem 40
3.2 Learning and Covering Problems 42
3.3 Greedy Algorithm and Analysis 43
3.4 Connection to Approximate Active Learning 45
3.5 Connection to Adaptive Submodularity 48
3.6 Negative Results for Simpler Approaches 51

3.7 Maximization Version 54
3.8 Application to Viral Marketing 56

Chapter 4: Noise and Implicit Hypothesis Class Expansion 59
4.1 Problem 60
4.2 Greedy Algorithm and Analysis 61
4.3 Connection to Noisy Active Learning 64
4.4 Interpretation as Implicit Hypothesis Class Expansion 71
4.5 Interpretation as Monotone Circuits of Constraints 72
4.6 Application to Movie Recommendation 74

Chapter 5: Batch Active Learning 79
5.1 Semi-Supervised Learning with Graph Cuts 80
5.2 Symmetric Submodular Functions as Regularizers 82
5.3 Active Semi-Supervised Learning Algorithm 85
5.4 Negative Results 91
5.5 Normalized Error Bound 93
5.6 Experiments 96

Chapter 6: Online Submodular Set Cover 102
6.1 Problem 103
6.2 Related Problems 105
6.3 Offline Analysis 107
6.4 Online Analysis 112
6.5 Experiments 116

Chapter 7: Conclusion 119

Bibliography 121

LIST OF FIGURES

Figure Number Page

2.1 Map of related problems. 19

3.1 Reductions to interactive submodular set cover. 41

4.1 Noisy interactive submodular set cover as a circuit. 73
4.2 Screenshot of the movie recommendation website. 75

5.1 A bad (left) and good (right) labeled set according to Ψ 84
5.2 Relative error. 97
5.3 Movies informative about taste. 100

6.1 Histograms used in offline analysis 108
6.2 E_i selects the ith element in S^t. 112
6.3 Average cover time 118

LIST OF TABLES

Table Number Page

3.1 Average number of queries to find a dominating set 56

4.1 Average survey responses. 77

ACKNOWLEDGMENTS

I'd like to thank my advisor Jeff Bilmes for his ideas, advice, and encouragement and for giving me the freedom to work on ideas that most interested me. I'd like to thank all of the researchers with whom I've had the pleasure of collaborating with or discussing research with including Amarnag Subramanya, Sumit Basu, Dan Morris, Ajit Singh, Erick Chastain, Hui Lin, Stefanie Jegelka, Rishabh Iyer, and many others. I'd like to thank my undergraduate research advisors, Tucker Balch, Charles Isbell, and Alexander Gray, who helped inspire me to attend graduate school. Finally, I'd like to thank Jessica Johnson for her love and support.

DEDICATION

For my parents

Chapter 1

INTRODUCTION

1.1 Motivation

Active learning is a variation of supervised learning in which the learning algorithm gets to pick which data points are labeled. In one standard setup, the learning algorithm is given a large pool of unlabeled examples. The algorithm then picks one-by-one examples to be labeled by a human labeler, using the labels from previous examples to choose which example is labeled next. The goal of the active learning algorithm is to learn a low error model using as few labeled examples as possible. Active learning is a popular topic in machine learning because in many real world problems unlabeled data is plentiful but labeled data is scarce. For example, consider labeling words in a document with their part-of-speech. It is very easy to acquire large quantities of unlabeled text, but it is comparatively hard to acquire labeled text. Active learning attempts to reduce dependence on labeled data by allowing the learner to choose informative or representative labeled points.

Submodular functions are a large class of set functions for which many otherwise intractable discrete optimization problems become easy to solve or approximate. Besides having nice theoretical properties, submodular functions also naturally arise in many real world applications. For this reason, submodular functions are sometimes viewed as a discrete version of convex functions; they strike a useful balance between theoretical and practical value.

Using submodular functions for active learning is an appealing idea for many reasons. Selecting a set of labeled points is inherently a discrete optimization problem. Moreover, greedy algorithms commonly used for active learning are similar to the greedy algorithms used in certain forms of submodular optimization. There is therefore hope that submodular functions could be a useful tool for analyzing active learning.

1.2 Background: Active Learning

1.2.1 Problem Setting Variations

The active learning problem setting we just outlined is an example of offline, adaptive active learning: the learning algorithm has access to a set of unlabeled points from which it queries for one label at a time, using previous labels to decide which example to label next. There are a number of different variations of the active learning problem which differ in their goals and assumptions. We outline some of these variations and define related terms.

Online vs Offline In an *online active learning* model the unlabeled data arrives in a stream and the active learning algorithm chooses whether or not to request a label for a data point before observing the next example [Dasgupta et al., 2005]. Online active learning methods are sometimes called *stream based*. In an *offline active learning* model the unlabeled data is instead a fixed size pool of examples all available at the same time. Greedy algorithms [Balcázar et al., 2007, Dasgupta, 2004, Tong and Koller, 2001] are offline: at each time step they select the best example in the pool. Offline active learning methods are sometimes called *pool based*.

Adaptive vs Batch An *adaptive* (or sequential, interactive) active learning algorithm chooses labeled examples one by one, using the labels of previously selected examples to choose which example to label next. Choosing examples in this way can in some cases give exponentially faster convergence than selecting examples all at once. Most work on active learning considers this kind of algorithm. *Batch* active learning algorithms instead select a set of examples to be labeled simultaneously. The motivation behind batch active learning is that in some cases it is cheaper to acquire a set of labels at once as opposed to a sequence of individual labels (the cost of labels is not additive). For example, if acquiring a label requires running a time consuming laboratory experiment, it may be significantly cheaper to run a set of experiments at once. Batch learning methods can be fully batch (select only a single set of examples) or partially batch (select more than one set of examples in series).

Realizable vs Agnostic Some active learning settings assume that the data set is *realizable*: that there is some classifier achieving zero error on the data. In contrast, in *agnostic active learning* [Balcan et al., 2006, Hanneke, 2007, Dasgupta et al., 2007] no

assumptions are made about noise or the error rate of the optimal classifier. There are also many models between agnostic and realizable. For example, we can assume low noise around the decision boundary [Balcan et al., 2007].

Statistical vs Adversarial In a *statistical learning* setting, the learning algorithm assumes that the data is generated according to some (usually unknown) distribution. Typically it is assumed that each unlabeled example is independently sampled from the same distribution (the data is *i.i.d.*, independent and identically distributed) [Balcan et al., 2006, Hanneke, 2007, Dasgupta et al., 2007, Beygelzimer et al., 2009, Balcan et al., 2007]. An alternative model is to consider *adversarial learning* (non-stochastic learning) where no assumption is made about the way in which the data is generated [Cesa-Bianchi et al., 2006, Orabona and Cesa-Bianchi, 2011]. In an adversarial setting, the data may be generated by an oblivious adversary in which case it is assumed that the data set is fixed in advance of the algorithm making any decisions. Alternatively the data may be generated by an adaptive adversary which can observe the decisions of the algorithm and use this information in generating future data points. Assuming an oblivious adversary can simplify the analysis, but for some applications an adaptive adversary is more realistic. For example, when predicting the price of a stock, the algorithm's predicted price could be used to make trading decisions which influence the future price of the stock.

Inductive vs Transductive In *inductive learning* the learning algorithm uses training data to produce some kind of classifier or model which is then evaluated on new, previously unseen test data. In *transductive learning* the learning algorithm is trained and evaluated on the same set of data points.

Parametric vs Non-parametric A *parametric* model is a model fully specified by a fixed size set of parameters. The size of a *non-parametric* model grows with the size of the data set. For example, a nearest-neighbors classifier is non-parametric.

Finite vs Inifinite Hypthoesis Class The *hypothesis class* is the set of classifiers under consideration by the learning algorithm. An example finite hypothesis class is the set of all decision trees with depth less than 5. An example infinite hypothesis class is the set of all linear classifiers with norm less than 1. Assuming a finite hypothesis class often simplifies analysis at the expense of reduced generality. Some methods which assume a

finite hypothesis class explicitly represent the hypothesis class as a finite set.

Single Round vs Repeated Most work on active learning considers a single learning problem. However, it's also interesting to consider *repeated active learning* where the learning algorithm must solve a sequence of related active learning problems. For example, consider a doctor who must diagnosis a sequence of patients. Each patient can be considered an instance of an active learning problem: the doctor must diagnosis the patient using as few tests as possible. Moreover, assuming the patients are all drawn from the same population, there may be similarities between the patients which the doctor can take advantage of to increase efficiency.

We note that these different axes of variation are far from orthogonal: certain sets of assumptions naturally go together while other groupings do not make sense. For example, inductive learning methods are almost always statistical–without making assumptions about the way in which the data is generated, its difficult to guarantee anything about performance on unseen data. Another example, in a transductive learning setting it often makes sense to assume a finite hypothesis class: if there is a finite set of allowed labels, then there are only finitely many ways to label a finite data set, so this assumption is without loss of generality.

1.2.2 Theory Based Approaches to Active Learning

Most theoretical work on active learning can be grouped into three broad clusters.

Inductive, Statistical Learning Work in this cluster typically assumes access to a stream of i.i.d. unlabeled data and learns an inductive classifier by querying for labels. A typical analysis is to then show a bound on the generalization error of the algorithm and a bound on the number of requested labels [Balcan et al., 2006, Hanneke, 2007, Dasgupta et al., 2007, Beygelzimer et al., 2009, Balcan et al., 2007]. Algorithms in this cluster often use a passive learning algorithm as a black box. These algorithms are usually general purpose in the sense that they can be used with any hypothesis class for which passive learning is possible. However, in order to show improvements over passive learning often additional assumptions are required concerning the hypothesis class, the way in which the data is generated, or both. An exception, it is possible to show *asymptotic* improvements in label

use under relatively few assumptions [Balcan et al., 2010].

Online, Adversarial Learning Work in this cluster assumes access to a stream of possibly non i.i.d. data. The learning algorithm makes a prediction for each unlabeled data point as it arrives. The learning algorithm then decides whether or not to query for the label of the data point. The goal of the algorithm is to make as few mistakes as possible (the setting is transductive). In this setting it is common to show a *regret guarantee*. These results are similar to passive online learning regret guarantees except that attention is also paid to the number of requested labels [Cesa-Bianchi et al., 2006, Orabona and Cesa-Bianchi, 2011]. Algorithms in this cluster usually assume a linear (or kernel) classifier model and often require assumptions about the true labels in order to show improvements. This work can be seen as the natural active learning extension of the online learning setting [Cesa-Bianchi and Lugosi, 2006] in much the same way that statistical active learning extends classic statistical learning theory.

Query Learning This third cluster generally refers to active learning using queries from a fixed finite set [Balcázar et al., 2007, Dasgupta, 2004, Hanneke, 2006]. The standard goal is to exactly identify a unique zero error hypothesis contained in a known finite hypothesis class. While this is in some ways more restrictive than other standard learning settings, the kinds of guarantees possible are much stronger. In this setting it is common to show a *competitive approximation ratio*: a guarantee that no other algorithm is much better than the proposed approach. Results of this kind can be either average case [Dasgupta, 2004] or worst case [Balcázar et al., 2007]. In an average case analysis there is assumed to be a distribution over the hypothesis space, and the goal is to show that on average over target hypotheses the learning algorithm performs well. Average case analyses avoid placing too much emphasis on hard to learn hypotheses. However, algorithms for the average case typically require that the prior distribution over hypotheses is known to the algorithm, which is not always the case. Worst case analyses of query learning are in some ways similar to online, adversarial learning in that no stochastic assumptions are made. However, query learning is very much offline: the entire set of queries is known up front.

Query learning is a less popular setting than statistical or online learning, and query learning results are sometimes viewed as impractical. Query learning is sometimes claimed

to be impractical because the learning algorithm is allowed to select any query and therefore may select queries which are very hard for a human to label [Settles, 2009]. However, this criticism is only valid if we set the query set to be, for example, all possible inputs to the classifier. An easy fix is to instead construct the query set from a finite set of unlabeled data. In this case it makes sense to use the *effective hypothesis class* [Dasgupta, 2004] for that data set: the hypothesis class produced by grouping together into equivalence classes hypotheses that label the data set identically. This ensures that the hypothesis class is finite and that if there is any zero-error hypothesis then it is unique.

A much larger barrier to practical adoption of query learning is that many query learning algorithms explicitly represent the hypothesis class. Recall that for transductive learning assuming the hypothesis class is finite is a natural assumption (usually without loss of generality). However, even if the hypothesis class is finite it is often too large to explicitly represent. Another major practical issue is the assumption of exact learning and zero noise. It is possible to relax these assumptions [Dasgupta et al., 2003, Balcázar et al., 2007], but these results are not very well known and their practical implications have not been explored. This is most likely because these algorithms assume a small, finite hypothesis class.

1.2.3 Heuristic Based Approaches

In this dissertation we are primarily concerned with theoretically justified learning methods. However, there is a large body of work considering heuristically motivated active learning methods that work well in practice but do not have rigorous theoretical guarantees. One such algorithm is the greedy margin based method of Tong and Koller [2001] for Support Vector Machine learning. This algorithm assumes a pool of unlabeled data. At each step it learns a linear classifier and then labels the unlabeled point closest to the decision boundary. While the method often works very well in practice, it does not have strong theoretical guarantees and can sometimes get stuck in locally optimal solutions [Schütze et al., 2006]. These heuristic methods sometimes have theoretically justified counterparts. For example, the analysis of Balcan et al. [2007] partially motivates greedy, margin based active learning. However, in general there are significant gaps between theory and practice. With the

exception of work on query learning, most theory based approaches are quite "mellow" and request labels for every example for which there is uncertainty. In contrast most heuristic based approaches operate in a pool based setting and are greedy. Settles [2009] gives a detailed survey of active learning including the many different heuristic methods that have been proposed. A primary goal in applying submodular optimization to active learning is to help bridge the gap between theory and practice.

1.3 Background: Submodular Functions

A set function $F(S) : 2^V \rightarrow \mathbb{R}$ maps subsets of some ground set V to real values. Throughout we will use $n = |V|$ to denote the size of the ground set. A set function $F(S)$ is called *submodular* iff for every $A \subseteq B \subseteq (V \setminus \{v\})$ we have

$$F(A + v) - F(A) \geq F(B + v) - F(B) \tag{1.1}$$

Intuitively, F exhibits diminishing returns: adding an element to a set A results in a larger gain than adding the same element to B, a superset of A. An equivalent definition is that for any $A \subseteq V$ and $B \subseteq V$

$$F(A) + F(B) \geq F(A \cup B) + F(A \cap B) \tag{1.2}$$

A function for which 1.1 (and 1.2) hold with equality everywhere is called *modular* (linear, additive). If a function $F(S)$ is submodular then $-F(S)$ is *supermodular*. If for every $A \subseteq B \subseteq V$ we have $F(A) \leq F(B)$ then F is called *monotone*. A set function is called *symmetric* if $F(A) = F(V \setminus A)$ for any $A \subseteq V$. A set function is normalized if $F(\emptyset) = 0$.

We review a number of standard examples and results. Proofs are omitted when they are simple or published in standard texts [Fujishige, 2005].

1.3.1 Examples

Graph Cut

Assume we are given a weighted, directed graph G with vertices V and non-negative edge weights specified by a non-negative weight matrix W (an edge weight of zero $W_{i,j} = 0$

denotes absence of an edge). Define the *graph cut* function $\Gamma(S)$ for $S \subseteq V$ to be the sum of edge weights for edges connecting $i \in S$ to $j \notin S$:

$$\Gamma(S) \triangleq \sum_{i \in S, j \notin S} W_{i,j} \tag{1.3}$$

Proposition 1.1. *The cut function $\Gamma(S)$ is submodular*

There are a number of basic optimization problems over graphs which can be defined in terms of $\Gamma(S)$. The *minimum cut* problem is to compute

$$\operatorname*{argmin}_{S \subset V : S \neq \emptyset} \Gamma(S)$$

The *minimum s-t cut* problem is to compute

$$\operatorname*{argmin}_{S \subseteq (V \setminus t) : s \in S} \Gamma(S)$$

where s and t are designated *source* and *sink* nodes. Both of these problems can be solved in polynomial time, for example, using maximum-flow [Dinic, 1970, Edmonds and Karp, 1972].

The *maximum cut* problem is to compute

$$\operatorname*{argmax}_{S \subseteq V} \Gamma(S)$$

Unlike minimum-cut, this problem is NP-hard, but has a constant factor approximation algorithm [Vazirani, 2001].

Set Cover

Assume we are given a set system (set of sets) $\mathcal{V} \subseteq 2^V$. Define the coverage function $F(\mathcal{S})$ for $\mathcal{S} \subseteq \mathcal{V}$ to be the cardinality of the union of all sets in $\mathcal{S}$.

$$F(\mathcal{S}) \triangleq |\bigcup_{S \in \mathcal{S}} S| \tag{1.4}$$

Proposition 1.2. *The set coverage function $F(\mathcal{S})$ is monotone submodular.*

The *set cover* problem is to compute the minimum cardinality subset of $\mathcal{V}$ that covers the ground set:

$$\operatorname*{argmin}_{\mathcal{S}\subseteq\mathcal{V}:F(\mathcal{S})=F(\mathcal{V})} |\mathcal{S}|$$

The *max k cover* problem is to compute the subset of cardinality k achieving maximum coverage:

$$\operatorname*{argmax}_{\mathcal{S}\subseteq\mathcal{V}:|\mathcal{S}|\leq k} F(\mathcal{S})$$

Both of these problems are NP-hard [Vazirani, 2001] but set cover has a $O(\log n)$ approximation algorithm and max k cover has a constant factor approximation algorithm. In particular, a simple greedy algorithm for set cover gives a solution with cost no more than $1 + \log n$ times greater than than the optimal solution. The same algorithm applied to max k cover gives a solution within $1 - 1/e$ times the optimal solution, and these ratios are the best possible under reasonable complexity assumptions [Feige, 1998].

Entropy and Mutual Information

Say that the ground set V is a set of (possibly correlated) random variables. For a set $S \subset V$ define $H(S)$ to be the entropy of the set of random variables S.

Proposition 1.3. *$H(S)$ is submodular.*

Define mutual information $I(A; B) = H(A) + H(B) - H(A \cup B)$.

Proposition 1.4. *$F(S) \triangleq I(S; V \setminus S)$ is symmetric and submodular.*

Maximization over Set Elements

Proposition 1.5. *$F(S) \triangleq \max_{s\in S} f(s)$ is submodular, monotone for any function $f(s)$ defined over $s \in V$*

This function is connected to the *facility location* problem.

1.3.2 Optimization Problems Involving Submodular Functions

Submodular functions have recently become a popular topic of study in applied and theoretical computer science because many optimization problems involving them are easy to solve or approximately solve.

Submodular function minimization is the problem of computing

$$\operatorname*{argmin}_{S \subseteq V} F(S)$$

for a submodular function $F(S)$. Submodular function minimization generalizes the minimum s-t cut problem. To see this, let $\Gamma(S)$ be the graph cut function and then define $F(S) \triangleq \Gamma((S \setminus \{t\}) \cup \{s\})$. It is not hard to show that $F(S)$ is submodular and minimizing $F(S)$ is equivalent to minimizing $\Gamma(S)$ under the constraint $s \in S$ and $t \notin S$.

Submodular function minimization can be solved in polynomial time [Grötschel et al., 1981, Iwata and Orlin, 2009]. For the special case of symmetric submodular functions (like graph cut) there is a $O(n^3)$ algorithm [Queyranne, 1998]. Note this algorithm finds a non-trivial minimizer (i.e. the set S minimizing $F(S)$ s.t. $S \neq \emptyset$ and $S \neq V$) and therefore strictly generalizes computing the minimum cut in a graph.

In its simplest form, *submodular function maximization* is the problem of computing

$$\operatorname*{argmax}_{S \subseteq V} F(S)$$

for a submodular objective $F(S)$. This problem generalizes maximum cut and is therefore NP-hard. However, as for max cut, there are constant factor approximation algorithms [Feige et al., 2007].

Of particular practical interest are maximization problems involving monotone submodular functions. In this case, the simplest, unconstrained version of the problem is trivial: the maximizing set is just the ground set V. However, constrained versions of the problem are non-trivial. The simplest constrained variation of submodular maximization is *cardinality constrained submodular function maximization*:

$$\operatorname*{argmax}_{S \subseteq V : |S| \leq k} F(S)$$

where k is a constant. This problem generalizes max k cover. As for max k cover, for monotone submodular objectives, the greedy algorithm gives an optimal $1 - 1/e$ approximation [Nemhauser et al., 1978]. There are several other kinds of constraints which are of interest including matroid constraints [Fisher et al., 1978, Calinescu et al., 2011] and knapsack constraints [Lee et al., 2010]. Cardinality constrained submodular maximization is in fact a special case of the matroid constrained problem.

Submodular set cover is the problem of computing

$$\operatorname*{argmin}_{S:F(S)=F(V)} c(S)$$

where $F(S)$ is a monotone submodular function and $c(S)$ is a nondecreasing modular cost function. This problem generalizes set cover. As for set cover, the greedy algorithm gives a log factor approximation [Wolsey, 1982]. In particular, if $F(V) = 1$ and the smallest nonzero gain of $F(S)$ is δ than the greedy algorithm gives a $1+\ln(1/\delta)$ approximation. We note a variation of this problem is to consider coverage constraints of the form $F(S) \geq \alpha$ where α is some specified partial coverage threshold. It is simple to reduce this version of the problem to the original by redefining the objective to be $\bar{F}(S) = \min(F(S), \alpha)$. $\bar{F}(S)$ is submodular, monotone when $F(S)$ is submodular, monotone and $\bar{F}(S) = \bar{F}(V)$ iff $F(S) \geq \alpha$.

Ranking with submodular valuations [Azar and Gamzu, 2011] is a recently proposed problem that generalizes submodular set cover. Define the *cover time* of a submodular function F with respect to a sequence $S = (v_1, v_2, \ldots v_n)$ to be

$$c(F, S) = \min_{i:F(S_i)=F(V)} i$$

where $S_i = \bigcup_{j \leq i} \{v_j\}$. This is the number of elements we need to achieve $F(S) = F(V)$ when we select elements in the order specified by S. In ranking with submodular valuations the goal is to find a sequence of ground set elements which approximately minimizes the average cover time of a collection of functions $F_1, F_2 \ldots F_T$

$$\min_{S \in V^n} \sum_{i=1}^{T} c(F_i, S)$$

Here V^n is defined to be the set of all sequences of length n (i.e. the Cartesian product of V with itself n times). Note that the sequence is allowed to contain duplicate items, but

because the objectives are set functions selecting an item twice has no benefit (at least one of the optimal solutions contains no duplicates). This can be viewed as a sort of averaged version of submodular set cover: submodular set cover with a uniform cost function is equivalent to ranking with submodular valuations with $T = 1$.

We use *online submodular optimization* to refer to repeated prediction versions of submodular optimization problems. For example, in online submodular minimization [Hazan and Kale, 2009, Jegelka and Bilmes, 2011], at each round the algorithm picks a set S_t and then a submodular function F_t is revealed and the algorithm pays cost $F_t(S_t)$. In this setting one reasonable goal of an online optimization algorithm is to perform nearly as well as the best fixed prediction in hindsight. In other words we want to ensure that

$$\sum_{i=1}^{T} F_t(S_t) - \min_{S \subseteq V} \sum_{i=1}^{T} F_t(S)$$

is small. This term is known as the *regret* of the algorithm. There also online versions of submodular maximization [Streeter and Golovin, 2008]. In Chapter 6 we discuss other online optimization problems and derive an algorithm for a new online version of submodular set cover.

In the *secretary setting* [Gupta et al., 2010b] (named after the famous secretary hiring problem) individual ground elements $v \in V$ are revealed one-by-one and the algorithm must decide immediately whether or not to include the current element in the final set S. Once an element is included in the set it cannot be removed.

1.3.3 Useful Properties of Submodular Functions

Part of what makes submodular functions theoretically and practically interesting is that, like convex functions, submodular functions can be transformed and combined in order to derive new submodular functions. We list a number of useful results of this kind here.

Proposition 1.6. *If $F(S)$ is submodular then the following are also submodular:*

- $G(S) \triangleq F(V \setminus S)$
- $G(S) \triangleq \min(F(S), c)$ *where c is a constant*

- $G(S) \triangleq cF(S)$ *where* c *is a non-negative constant*
- $G(S) \triangleq \min_{T \subseteq S} F(T)$
- $G(S) \triangleq \min_{T \supseteq S} F(T)$
- $G(S) \triangleq F(S \cap T)$ *where* T *is some fixed set*
- $G(S) \triangleq F(S \cup T)$ *where* T *is some fixed set*

Proposition 1.7. *If* $F_i(S)$ *for* $i = 1 \dots m$ *are all submodular then* $G(S) = \sum_{i=1}^{m} F_i(S)$ *is submodular.*

Submodular functions are related to both convex and concave functions [Lovász, 1983]. In particular every submodular function has an extension to $\mathbb{R}^n$ which is convex, called the *Lovász extension*. Before defining the extension, we first define the *submodular polyhedron* for a submodular function F with $F(\emptyset) \geq 0$.

$$P_F = \{x \in \mathbb{R}^V : x \geq 0, \quad x(A) \leq F(A) \ \ \forall A \subseteq V\}$$

Here we are using $x \in \mathbb{R}^V$ as a modular function. $x(A)$ denotes $\sum_{s \in A} x(s)$. Consider the following linear program for any $x \in \mathbb{R}^V$

$$\max_{w \in P_F} w^T x \tag{1.5}$$

A celebrated result of Edmonds [1970] shows this linear program can be solved in polynomial time via a greedy algorithm similar to the greedy algorithm for computing a maximum weight independent set in a matroid

Theorem 1.1 ([Edmonds, 1970])**.** *Given a vector* $x \in \mathbb{R}^V$*, define an ordering* $j_1 \dots j_n$ *such that* $x(j_1) \geq x(j_2) \dots \geq x(j_k) > 0$*. The following vector* w *gives the solution to (1.5)*

$$\begin{aligned} w(j_1) &= F(A_1) \\ w(j_i) &= F(A_i) - F(A_{i-1}) && \text{for } 2 \leq i \leq k \\ w(j_i) &= 0 && \text{for } k < i < n \end{aligned}$$

where $A_i = \{j_1, \ldots j_i\}$.

For a set function F define the Lovász extension to be

$$f(x) = w^T x$$

where w is the result of the greedy algorithm applied to x and F. It is clear from Edmonds [1970] that $f(x)$ is convex when F is submodular: in this case an equivalent definition is $f(x) = \max_{w \in P_F} w^T x$, which is the maximum of linear functions. Lovász [1983] shows the extension defined by the greedy algorithm is in fact convex *only* if F is submodular.

Theorem 1.2 ([Lovász, 1983])**.** *$f(x)$ is convex iff $F(S)$ is submodular.*

The Lovász extension relates submodular functions to convex functions. On the other hand (1.1) makes submodular functions seem more similar to concave functions. In fact we have the following

Proposition 1.8. *A function $F(S) = f(|S|)$ is submodular if f is concave.*

1.3.4 Practical Applications of Submodular Optimization

Submodular functions are of practical interest because diminishing returns and economies of scale arise naturally in many real world problems.

Clustering can be posed as a submodular function minimization problem [Narasimhan et al., 2006]. Similarly many forms of image segmentation are special cases of submodular minimization, including forms more general than graph cut [Stobbe and Krause, 2010, Kohli et al., 2009]. In image segmentation, the submodular function typically measures the cost of separating a set of pixels from the rest of the image. The goal is then to extract a subset of pixels from the image with minimal separation cost. The Lovász extension of a submodular function can be used as a regularization term when learning a linear model in order to enforce particular structured sparsity patterns [Bach, 2010]. In this case certain related proximal optimization problems involve submodular minimization.

Submodular maximization and submodular set cover share many applications: submodular set cover can be viewed as a sort of dual of submodular maximization where the goal is

not to maximize quality subject to an upper bound on cost but rather minimize cost subject to a lower bound on quality. The influence of a set of nodes in a social network is a submodular function under several models of influence [Kempe et al., 2005]. These objectives are useful for applications like viral marketing. Many kinds of sensor placement problems can be modeled as submodular maximization or set cover problems [Krause et al., 2008b,a]. In these problems the submodular function measures the coverage achieved by placing sensors at a set of locations. Extractive document summarization problems [Lin and Bilmes, 2010, 2011] can also be modelled as submodular maximization problems; here the submodular function computes the quality of a set of sentences. Online submodular maximization can be applied to task scheduling [Streeter and Golovin, 2008] and to deciding which sensors in a network to activate in a repeated sensing task [Golovin et al., 2010a].

1.4 A Simple Approach

The previous section, in particular the discussion of sensor placement, suggests a simple approach to active learning using submodular optimization: we can pose selecting the labeled set as a submodular maximization or submodular set cover problem. If we set set V to be the set of all unlabeled data points and $F(S)$ to be some measure of "informativeness" (as in the sensor placement application), then cardinality constrained submodular maximization corresponds to choosing a maximally informative labeled set within a budget constraint. Similarly, submodular set cover corresponds to picking a minimal cost labeled set that achieves some desired level of informativeness. This approach has in fact been proposed and evaluated empirically [Hoi et al., 2006, Shi et al., 2010, Joshi et al., 2010] for several different submodular objective functions. Unfortuantely there are some theoretical and practical problems with this approach.

First, because standard submodular optimization methods will evaluate the gain of $F(S)$ on every $v \in V$, $F(S)$ can only depend on the set of unlabeled points–not on the observed labels for the points we select. If $F(S)$ were to depend on the observed labels then we'd need to know the labels for every $v \in V$ to run submodular maximization, which defeats the purpose of active learning. Therefore, the straightforward application of this approach necessarily produces a batch active learning method as opposed to an adaptive active learning

method. Batch active learning has strengths, but we would ideally like to apply this kind of approach even in adaptive settings.

A second problem is that we must define some submodular objective which measures informativeness. There are many reasonable such objectives, but it is not clear why to prefer one over the other. Empirical evaluations can guide our choice of objective, but performance may vary depending on the application or the choice of learning algorithm.

A third, related problem is that it is not clear how to relate the theoretical guarantees of the submodular optimization method back to the original learning problem. If we define some submodular objective which measures informativeness and then apply submodular maximization, we are guaranteed to find a set which approximately maximizes our chosen measure of informativeness. However, we are not necessarily guaranteed that the resulting labeled set is useful for the original learning problem. For example, we do not necessarily have any guarantee that the classifier we learn will have low error.

1.5 Our Contributions

We investigate approaches which go beyond the simple approach to active learning through submodular optimization. In Chapters 2, 3, and 4 we propose interactive versions of submodular optimization with application to active learning. Chapter 2 begins with very simple interactive versions of submodular set cover and submodular maximization. The proceeding two chapters extend and generalize the simple problems to incorporate hypothesis classes, multiple objectives, and noise. With appropriate choice of the submodular objective functions we can recover and generalize several query learning results including results for approximate and noisy learning. Of particular practical interest, in Chapter 4 we show that in certain cases we can use submodular functions to implicitly represent or expand hypothesis classes. This works towards addressing one of the major practical difficulties with query learning: the need for an explicit hypothesis class. In Chapter 5 we investigate a complimentary batch learning setting. In this setting we uses submodular functions not as objectives to be maximized but as regularization terms to be minimized. We derive error bounds using symmetric submodular regularization and give active semi-supervised learning algorithms that approximately minimize these bounds. In Chapter 6 we

examine repeated active learning. We show that certain repeated active learning problems can be treated as an online version of submodular set cover.

A common theme in this work is a focus on learning settings that are very different from the standard settings. The results in Chapters 2, 3, 4, and 5 are for settings which are adversarial but also offline. The repeated learning setting of Chapters 6 is also unusual. The more general contribution of this work is therefore evidence that submodular optimization is a useful tool for expanding the scope of learning theory.

Portions of Chapters 2, 3, and 4 were originally presented in two conference papers:

- A. Guillory and J. Bilmes. Interactive submodular set cover. In *ICML*, 2010a
- A. Guillory and J. Bilmes. Simultaneous learning and covering with adversarial noise. In *ICML*, 2011b

The former was also published as an extended tech report:

- A. Guillory and J. Bilmes. Interactive submodular set cover, 2010b. Technical Report. UWEETR-2010-0001

And the later was originally published in a workshop:

- A. Guillory and J. Bilmes. Simultaneous learning and covering with adversarial noise. In *NIPS 2010 Workshop on Discrete Optimization in Machine Learning (DISCML)*, 2010c

Chapter 5 was originally published in two conference papers:

- A. Guillory and J. Bilmes. Label Selection on Graphs. In *NIPS*, 2009
- A. Guillory and J. Bilmes. Active Semi-Supervised Learning using Submodular Functions. In *UAI*, 2011c

Chapter 6 was originally published in the following conference paper:

- A. Guillory and J. Bilmes. Online Submodular Set Cover, Ranking, and Repeated Active Learning. In *NIPS*, 2011a

Chapter 2

INTERACTIVE SUBMODULAR OPTIMIZATION

Recall that the "simple approach" to active learning through submodular optimization is to set our ground set V to be an unlabeled data set, define some submodular notion of informativeness $F(S)$, and then apply some black box submodular maximization or submodular set cover algorithm. A problem with this approach is that it necessarily produces a batch active learning algorithm which does not use information from previously selected labels in deciding which point to label next. This is because any standard submodular maximization or submodular set cover algorithm will evaluate $F(S)$ on sets containing every ground set element. Therefore our objective $F(S)$ cannot assume access to the labels for S when evaluating $F(S)$. If we assumed access to this information then to run a black box submodular optimization method we'd need to label every data point, defeating the purpose of active learning. If $F(S)$ does not incorporate information about the labels of S then the resulting active learning algorithm is batch.

In this chapter we seek answers to the question: how can we make submodular optimization interactive? We begin by defining two simple problems: interactive submodular set cover and interactive submodular maximization. We give greedy algorithms and analyses for these problems. It turns out that interactive submodular set cover generalizes exact active learning with a finite query set [Balcázar et al., 2007]. Through this relationship we will derive approximation guarantees for active learning. Moreover, we will show that the relationship between interactive submodular set cover and submodular set cover parallels that of exact active learning and set cover. We will show submodular set cover is the passive version of interactive submodular set cover in the same way that set cover is the passive version of exact active learning. Figure 2.1 illustrates this.

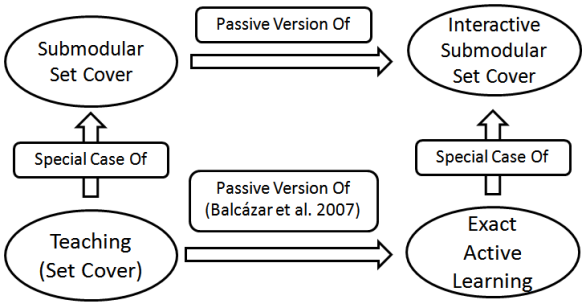

Figure 2.1: Map of related problems.

2.1 Problems

Below we define a simple, interactive version of submodular set cover.

Interactive Submodular Set Cover (Simple Version)

Given:

- Query set Q and response set R
- Modular, positive query cost function c defined over Q
- Submodular, monotone function $F : 2^{Q \times R} \to \mathbb{R}_{\geq 0}$

Protocol: At step i algorithm asks $q_i \in Q$, receives a response $r_i \in R$ from nature

Goal: Achieve $F(\bigcup_{j \leq i} \{(q_j, r_j)\}) = F(Q \times R)$ using minimal cost $\sum_{j \leq i} c(q_i)$

This problem is similar to the standard submodular set cover problem. As in the standard problem, the goal is to achieve a coverage constraint using minimal cost. However, unlike the standard problem, here $F(S)$ is not defined over 2^V but rather sets of question-response pairs $2^{Q \times R}$. Moreover, the optimization algorithm operates according to a protocol where at step i the algorithm selects a question $q_i \in Q$ but the corresponding response $r_i \in R$ is *not* selected by the algorithm but rather by nature. In this way the problem is interactive. The question asking continues until the first time step i^* where $F(\bigcup_{j \leq i^*} \{(q_j, r_j)\}) = F(Q \times R)$. The cost incurred by the algorithm is then $\sum_{j \leq i^*} c(q_j)$.

By making the problem interactive we overcome the limitation of the simple approach discussed in the introduction to this chapter. Consider setting Q to be a set of unlabeled data, R a set of allowed labels, and F some measure of the quality of a partially labeled set $S \subseteq (Q \times R)$. Solving the interactive submodular set cover problem corresponds to *adaptively* selecting a good, minimal cost labeled set. Note that we do not assume that we know a priori the labels for the data points. The label corresponding to a data point is only revealed after we have made the irrevocable decision to add a data point to the labeled set. We do however assume that we know $F(S)$, the measure of quality through which partially labeled sets are judged. In future sections we will show that certain active learning problems can be reduced to interactive submodular set cover through specific choices of $F(S)$. Through these reductions we give approximation results for active learning.

We can similarly define an interactive version of cardinality constrained submodular maximization.

Interactive Submodular Maximization

Given:

- Query set Q and response set R
- Submodular, monotone function $F : 2^{Q \times R} \rightarrow \mathbb{R}_{\geq 0}$
- Integer k

Protocol: At step i algorithm asks $q_i \in Q$, receives a response $r_i \in R$ from nature

Goal: Maximize $F(\bigcup_{i=1}^{k} \{(q_i, r_i)\})$

The difference between this problem and interactive submodular set cover is that here we ask a fixed number of questions k. The goal is no longer to achieve a particular coverage constraint but rather to maximize the resulting objective value after asking k questions.

2.2 Worst Case Cost and Value

It is not immediately obvious how to evaluate algorithms for interactive submodular set cover or interactive submodular maximization. One approach would be to assume nature behaves according to some probabilistic model. For example one very simple model would be to

assume nature chooses each response uniformly at random from R. We could then analyze the expected cost of an algorithm for interactive submodular set cover or the expected objective value of an algorithm for interactive submodular maximization. A problem with this approach is that for many applications it is hard to come up with a realistic probabilistic model. We instead consider an adversarial setting and analyze performance with respect to worst case responses.

An instance of the interactive submodular set cover problem is defined by its inputs (Q, R, F, c). We say that an algorithm is *correct* for (Q, R, F, c) if it continues asking questions until $F(S_i) = F(Q \times R)$. The cost incurred by a particular execution of an algorithm is $\sum_{j<i} c(q_j)$. Define the *worst case cost* of an algorithm on (Q, R, F, c) to be the largest possible cost incurred by the algorithm when run on that problem instance, maximizing over all possible sequences of responses. We allow for algorithms to use randomization, but in this case we still define the worst case cost to be the largest possible cost incurred (the maximum cost that occurs with non-zero probability). It is therefore not clear that randomization can help. Note that the sequence of responses which causes an algorithm to incur maximum cost may differ from algorithm to algorithm. In this sense worst case cost is with respect to an adversary with knowledge of our algorithm.

Some instances of interactive submodular set cover have infinite worst case cost for every algorithm. More precisely, there are instances for which no correct algorithm always terminates. This is the case if it is possible for nature to respond so that even after we have asked every question $F(S) < F(Q \times R)$. We will always analyze the cost of an algorithm on a particular problem instance with respect to the cost of the optimal algorithm for that same instance. Therefore we will not need to treat these infinite cost problems specially.

An instance of interactive submodular maximization is specified by (Q, R, F, k). We say an algorithm for interactive submodular maximization is correct for (Q, R, F, k) if it asks k questions. The value achieved by an execution of an algorithm is $F(S_k)$. Define the *worst case value* of an algorithm on (Q, R, F, k) to be the smallest possible objective value achieved by the algorithm when run on that problem instance, minimizing over all possible sequences of responses.

Note that we have defined the worst case cost and worst case value of an algorithm

specific to a particular problem instance. Our analyses will also be problem instance specific. We will present an algorithm which for any interactive submodular set cover problem (Q, R, F, c) has worst case cost not much more than that of any other algorithm for that problem instance. Similarly we will present an algorithm that for any interactive submodular maximization problem (Q, R, F, k) has worst case value not much less than that of any other algorithm for that problem instance.

This kind of guarantee is much stronger than, for example, analyzing the worst case cost of an algorithm for worst case problem instances. To see this recall that for interactive submodular set cover there are problem instances for which every algorithm has infinite worst case cost. Therefore an instance specific analysis is actually necessary to give non-trivial worst-case guarantees. To make this distinction more clear, say that the query set Q is a set of unlabeled data and solving (Q, R, F, c) corresponds to solving an active learning problem on that data set (we will give examples of this in future sections). Our algorithm is guaranteed to solve the active learning problem with approximately minimal cost *on that specific data set*. In other words, no other correct algorithm can achieve much better cost on that specific data set, including algorithms tailored to the specific data set. This is despite the fact that our algorithm is generic and can be applied to any data set.

2.3 Greedy Algorithm

For the non-interactive submodular set cover and cardinality constrained submodular maximization problems a simple greedy algorithm gives optimal approximation ratios. More specifically the greedy algorithm for submodular set cover starts with the empty set $S_0 \leftarrow \emptyset$. At step i it then adds to S_{i-1} the ground set element with the largest cost-normalized objective gain.

$$\operatorname*{argmax}_{s \in V} \frac{F(S_{i-1} + s) - F(S_{i-1})}{c(s)}$$

Wolsey [1982] show this algorithm gives a $1 + \ln(F(V)/\delta)$ approximation when the smallest non-zero gain of $F(S)$ is δ. For cardinality constrained submodular maximization, the algorithm is identical except that the "cost" of each ground set element is the same so the

algorithm simplifies to

$$\operatorname*{argmax}_{s \in V} F(S_{i-1} + s)$$

Nemhauser et al. [1978] show this algorithm achieves a $1 - 1/e$ approximation.

It makes sense to also consider greedy approaches for the interactive problem. However, the difficulty is that in the interactive problem the gain of asking a question depends on the response we receive. Since we don't control the responses we receive, we cannot predict the gain of a question until after we have asked it. The approach we take is to consider the *worst case gain* of a question. Define the *worst case greedy* algorithm for interactive submodular set cover to be the algorithm which at time step i selects the question with maximal worst case gain

$$\operatorname*{argmax}_{q \in Q} \min_{r \in R} \frac{F(S_{i-1} + (q, r)) - F(S_{i-1})}{c(q)}$$

Here $S_i = \bigcup_{j \leq i} \{(q_j, r_j)\}$ is the set of question response pairs observed through time step i. The full algorithm is shown in Algorithm 2.1.

In some cases it may be infeasible to implement the worst case greedy algorithm. This may be because there are a very large number of queries or because it is hard to evaluate F exactly. In this case it may still be possible to approximately implement the algorithm. Define a $(1 - \epsilon)$ approximate greedy strategy to be any algorithm which selects a question q with

$$\min_{r \in R} \frac{F(S_{i-1} + (q, r)) - F(S_{i-1})}{c(q)} \geq (1 - \epsilon) \max_{q \in Q} \min_{r \in R} \frac{F(S_{i-1} + (q, r)) - F(S_{i-1})}{c(q)}$$

We will show that if the greedy algorithm is only approximate then the optimality degrades gracefully with $(1 - \epsilon)$.

We can similarly define the worst case greedy algorithm for interactive submodular maximization to be the algorithm which chooses

$$\operatorname*{argmax}_{q \in Q} \min_{r \in R} F(S_{i-1} + (q, r))$$

The algorithm is shown in Algorithm 2.2. Note if $c(q) = 1$ for every $q \in Q$ than Algorithm 2.1 and Algorithm 2.2 are identical except for their stopping criteria.

Algorithm 2.1 Worst Case Greedy for Interactive Submodular Set Cover

$S_0 \leftarrow \emptyset$

$i = 1$

while $F(S_{i-1}) < F(Q \times R)$ **do**

$q_i \leftarrow \underset{q \in Q}{\text{argmax}} \ \min_{r \in R} \frac{F(S_{i-1}+(q,r))-F(S_{i-1})}{c(q)}$

Ask q_i and receive response r_i

$S_i \leftarrow S_{i-1} + (q_i, r_i)$

$i \leftarrow i + 1$

end while

Algorithm 2.2 Worst Case Greedy for Interactive Submodular Maximziation

$S_0 \leftarrow \emptyset$

$i = 1$

while $i \leq k$ **do**

$q_i \leftarrow \underset{q \in Q}{\text{argmax}} \ \min_{r \in R} F(S_{i-1} + (q, r))$

Ask q_i and receive response r_i

$S_i \leftarrow S_{i-1} + (q_i, r_i)$

$i \leftarrow i + 1$

end while

For interactive submodular maximization define a $(1 - \epsilon)$ approximate greedy strategy to be any algorithm which selects a question q with

$$\min_{r \in R}((S_{i-1} + (q, r)) - F(S_{i-1})) \geq (1 - \epsilon) \max_{q \in Q} \min_{r \in R}(F(S_{i-1} + (q, r)) - F(S_{i-1}))$$

This is again identical to the algorithm for the min-cost coverage version of the problem except that costs are uniform.

2.4 Analysis

Because the problem is interactive, the standard proofs for analyzing the greedy algorithms for submodular set cover and cardinality constrained submodular maximization do not

apply. Our analysis instead draws from worst case analyses of query learning [Hanneke, 2006, Balcázar et al., 2007]. We begin by analyzing the set cover version of the problem.

Define an *oracle* to be a function mapping questions to responses $T : Q \to R$. As a short hand we will use $T(\hat{Q})$ where $\hat{Q}$ is a set of questions to denote the set of question-response pairs induced by T and $\hat{Q}$.

$$T(\hat{Q}) \triangleq \bigcup_{q \in \hat{Q}} \{(q, T(q))$$

We now define General Cover Cost

$$GCC(Q, R, F, c) = \max_{T \in R^Q} \min_{\hat{Q} \subseteq Q : F(T(\hat{Q})) = F(Q \times R)} c(\hat{Q})$$

Where the arguments are clear from context, we write $GCC(Q, R, F, c)$ as simply GCC. GCC can be thought of as as the worst-case cost needed to achieve $F(S) = F(Q \times R)$ *when the algorithm knows how questions will be answered.* Note that in fact the interior minimization operator in GCC is solving a standard submodular set cover problem with $V = T(Q)$.

For certain problem instances there is an oracle T such that even after asking every question $q \in Q$, $F(S) = F(T(Q)) < F(Q \times R)$. In this case we define GCC to be infinity. It is not hard to see that these problems are exactly the problem instances for which interactive submodular set cover has infinite worst case cost.

GCC is analogous to the Teaching Dimension [Hanneke, 2006, Balcázar et al., 2007] and similar quantities used to analyze query learning. The main difference is that while Teaching Dimension is defined in terms of a set cover problem, GCC is defined in terms of a submodular set cover problem. The relationship between submodular set cover and interactive submodular set cover is in some sense parallel to that of set cover and query learning. We show that $GCC(Q, R, F, c)$ lower bounds the worst case cost of any algorithm on (Q, R, F, c). We then show that the greedy algorithm's cost when run on (Q, R, F, c) is approximately upper bounded by this same quantity. These two results combine to show the approximate optimality of the greedy algorithm for any problem instance. This mirrors the analysis of greedy query learning using the Teaching Dimension.

Lemma 2.1. *If there is an algorithm which is correct on the interactive submodular set*

cover instance (Q, R, F, c) *with finite worst case cost* C^* *then* $GCC(Q, R, F, c) \leq C^*$

Proof. Assume the lemma is false and there is a correct algorithm with worst case cost $C^* < GCC(Q, R, F, c)$. Let T^* be the oracle achieving maximum cost in the definition of GCC

$$T^* = \operatorname*{argmax}_{T \in R^Q} \min_{\hat{Q} \subseteq Q : F(T(\hat{Q})) = F(Q \times R)} c(\hat{Q})$$

Execute the algorithm using T^* to answer questions. Since the algorithm is correct it will ask questions until $F(S_i) = F(Q \times R)$. Let Q_i be the set of questions asked by the algorithm. Since we used the oracle to answer questions $S_i = T^*(Q_i)$ and $F(T^*(Q_i)) = F(Q \times R)$. Since the algorithm has worst case cost C^* we know $c(Q_i) \leq C^*$. We now have a contradiction since from the definition of GCC we have

$$c(Q_i) \leq C^* < GCC = \min_{\hat{Q} \subseteq Q : F(T^*(\hat{Q})) = F(Q \times R)} c(\hat{Q})$$

That is, the cost of Q_i is strictly less than the minimum cost set of questions $\hat{Q}$ needed to ensure $F(T^*(\hat{Q})) = F(Q \times R)$. □

We now show that when GCC is small then the worst-case greedy algorithm always makes progress: there is always some question which for worst case response increases F.

Lemma 2.2. *For any set of questions response pairs* $S \subseteq (Q \times R)$ *there must be a question* $q \in Q$ *such that*

$$\min_{r \in R} \frac{F(S + (q, r)) - F(S)}{c(q)} \geq \frac{F(Q \times R) - F(S)}{GCC}$$

Proof. Assume the lemma is false and there is some $S \subseteq (Q \times R)$ such that for every question $q \in Q$ there is an $r \in R$ with

$$\frac{F(S + (q, r)) - F(S)}{c(q)} < \frac{F(Q \times R) - F(S)}{GCC}$$

Define an oracle T^* which answers questions with responses satisfying this inequality. In other words

$$T^*(q) \triangleq \operatorname*{argmin}_{r \in R} \frac{F(S + (q, r)) - F(S)}{c(q)}$$

From the definition of GCC

$$\min_{\hat{Q} \subseteq Q : F(T^*(\hat{Q})) = F(Q \times R)} c(\hat{Q}) \leq GCC$$

so we know there is a sequence of questions $\hat{Q}$ with $c(\hat{Q}) \leq GCC$ and $F(T^*(\hat{Q})) = F(Q \times R)$. Because F is monotone, $F(T^*(\hat{Q}) \cup S) = F(Q \times R)$. Using the submodularity of F

$$\begin{aligned} F(T^*(\hat{Q}) \cup S) &\leq F(S) + \sum_{q \in \hat{Q}} F(S + (q, T^*(q))) - F(S) \\ &< F(S) + \sum_{q \in \hat{Q}} \frac{c(q)(F(Q \times R) - F(S))}{GCC} \\ &\leq F(Q \times R) \end{aligned}$$

which is a contradiction. □

We are now prepared to upper bound the cost of the (approximate) worst case greedy algorithm in terms of GCC.

Lemma 2.3. *Let the cost incurred by a $(1 - \epsilon)$ approximate worst case greedy algorithm executed on interactive submodular set cover instance (Q, R, F, c) be C.*

$$C \leq GCC(Q, R, F, c) \frac{1}{(1 - \epsilon)} (1 + \ln \frac{F(Q \times R)}{\delta})$$

where δ is the smallest non-zero gain of F.

Proof. Let q_i be the ith question asked and S_i be the set of question-response pairs after asking q_i as in the description of the algorithm (Algorithm 2.1). Define C_i to be the cost accumulated up through step i

$$C_i = \sum_{j \leq i} c(q_i)$$

By Lemma 2.2 and the definition of $(1 - \epsilon)$ approximate greedy we have

$$\frac{F(S_i) - F(S_{i-1})}{c(q)} \geq (1 - \epsilon) \frac{F(Q \times R) - F(S_{i-1})}{GCC}$$

After some algebra we have

$$F(Q \times R) - F(S_i) \leq (F(Q \times R) - F(S_{i-1}))(1 - \frac{(1 - \epsilon)c(q_i)}{GCC})$$

Now we use $1 - x \leq e^{-x}$

$$F(Q \times R) - F(S_i) \leq (F(Q \times R) - F(S_{i-1}))e^{-(1-\epsilon)c(q_i)/GCC}$$

We can then apply this bound recursively and use the fact that $F(Q \times R) - F(S_0) \leq F(Q \times R)$ to get

$$F(Q \times R) - F(S_i) \leq F(Q \times R)e^{-(1-\epsilon)C_i/GCC}$$

In other words, the gap between $F(S_i)$ and $F(Q \times R)$ decreases exponentially with the cost incurred.

Let j be the largest integer such that

$$F(Q \times R) - F(S_j) \geq \delta$$

holds. We know

$$\delta \leq F(Q \times R)e^{-(1-\epsilon)C_j/GCC}$$

Solving for C_j we have

$$C_j \leq GCC\frac{1}{(1-\epsilon)}\ln(F(Q \times R)/\delta)$$

By the definition of j

$$F(Q \times R) - F(S_{j+1}) < \delta$$

However, the smallest non-zero gain of F is δ, so we in fact must have $F(S_{j+1}) = F(Q \times R)$. In other words, q_{j+1} is the last question asked. By Lemma 2.2 we know that the cost of this final question is at most $GCC\frac{1}{(1-\epsilon)}$, completing the proof. □

Combining Lemma 2.1 and Lemma 2.3 we have the following result, showing the approximate optimality of the worst-case greedy algorithm for interactive submodular set cover.

Theorem 2.1. *If there is an algorithm which is correct on the interactive submodular set cover instance (Q, R, F, c) with finite worst case cost C^* then a $\frac{1}{(1-\epsilon)}$ worst case greedy algorithm incurs cost C with*

$$C \leq C^*\frac{1}{(1-\epsilon)}(1 + \ln\frac{F(Q \times R)}{\delta})$$

where δ is the smallest non-zero gain of F.

We now turn our attention to the maximization version of the problem, the analysis for which follows nicely from the set cover analysis.

Lemma 2.4. *Assume there is a correct algorithm for interactive submodular maximization instance (Q, R, F, k) achieving worst case value α^*. Define $F_{\alpha^*}(S) = \min(F(S), \alpha^*)$ and let c be uniform, $c(q) = 1$ for all $q \in Q$.*

$$GCC(Q, R, F_{\alpha^*}, c) \leq k$$

Proof. First note that F_{α^*} is submodular, monotone, and non-negative (Proposition 1.6). The algorithm achieving worst case value α^* in k steps is also an algorithm for the interactive submodular set cover problem (Q, R, F_{α^*}, c) with worst case cost k. The Lemma then follows from Lemma 2.1. □

Lemma 2.5. *Assume there is a correct algorithm for interactive submodular maximization instance (Q, R, F, k) achieving worst case value α^*. Let α be the value achieved by a $(1-\epsilon)$ approximate worst case greedy algorithm on this instance.*

$$\alpha \geq (1 - e^{-(1-\epsilon)k/GCC(Q,R,F_{\alpha^*},c)})\alpha^*$$

where $F_{\alpha^*}(S) = \min(F(S), \alpha^*)$ and $c(q) = 1$ for all $q \in Q$.

Proof. As before, let q_i be the ith question asked by the worst-case greedy algorithm and S_i be the set of question-response pairs after asking q_i. Note that questions which maximize the worst-case gain of $F(S)$ also maximize the worst case gain of $F_{\alpha^*}(S)$. Therefore by Lemma 2.2 we have

$$F_{\alpha^*}(S_i) - F_{\alpha^*}(S_{i-1}) \geq (1-\epsilon)\frac{F_{\alpha^*}(Q \times R) - F_{\alpha^*}(S_{i-1})}{GCC(Q, R, F_{\alpha^*}, c)}$$

Because we have assumed there is an algorithm achieving worst-case value α^*, $F_{\alpha^*}(Q \times R) = \alpha^*$. Applying this

$$F_{\alpha^*}(S_i) - F_{\alpha^*}(S_{i-1}) \geq (1-\epsilon)\frac{\alpha^* - F_{\alpha^*}(S_{i-1})}{GCC}$$

After some algebra we have

$$\alpha^* - F_{\alpha^*}(S_i) \leq (\alpha^* - F_{\alpha^*}(S_{i-1}))(1 - \frac{1-\epsilon}{GCC})$$

Now we use $1 - x \leq e^{-x}$ to get

$$\alpha^* - F_{\alpha^*}(S_i) \leq (\alpha^* - F_{\alpha^*}(S_{i-1}))e^{-(1-\epsilon)/GCC}$$

Finally we can apply the bound recursively starting from k to get

$$\alpha^* - F_{\alpha^*}(S_k) \leq \alpha^* e^{-(1-\epsilon)k/GCC}$$

$\alpha^* - F(S_k) \leq \alpha^* - F_{\alpha^*}(S_k)$ so we have the desired bound. □

Combining Lemma 2.4 and Lemma 2.5 we have the following.

Theorem 2.2. *If there is an algorithm for interactive submodular maximization instance (Q, R, F, k) achieving worst case value α^* then a $(1-\epsilon)$ approximate worst case greedy algorithm achieves value α with*

$$\alpha \geq (1 - (\frac{1}{e})^{1-\epsilon})\alpha^*$$

2.5 Explicit Coverage Threshold

A straightforward extension is to alter the interactive submodular set cover problem to include an explicit coverage threshold. In other words, replace the goal $F(S) = F(Q \times R)$ with a goal $F(S) \geq \alpha$. We can reduce this version of the problem to the original by defining $F_\alpha(S) = \min(F(S), \alpha)$. This function is still submodular and monotone (Proposition 1.6). Assuming there is some S such that $F(S) \geq \alpha$ then $F(S) \geq \alpha$ iff $F_\alpha(S) = F_\alpha(Q \times R)$. Note that with this reduction the δ in Theorem 2.1 is changed to the smallest non-zero gain of F_α (which may be different than the smallest non-zero gain of F). We suspect however that a more direct analysis can also show the same result with δ the smallest non-zero gain of F.

Corollary 2.1. *If there is an algorithm for achieving $F(S) \geq \alpha$ using finite worst case cost C^* then a $\frac{1}{(1-\epsilon)}$ worst case greedy algorithm incurs cost applied to $F_\alpha(S) = \min(F(S), \alpha)$ with*

$$C \leq C^* \frac{1}{(1-\epsilon)}(1 + \ln \frac{\alpha}{\delta})$$

where δ is the smallest non zero gain of $F_\alpha(S)$

2.6 Restricted Adversary

In the analyses presented so far we have allowed nature to respond arbitrarily. In many applications it makes sense to place restrictions on nature. Consider an active learning problem where Q is a set of unlabeled points and R is a set of labels. Allowing nature to respond arbitrarily corresponds to allowing the unlabeled set to be labeled arbitrarily. For many applications, this is overly pessimistic, and it is reasonable to assume some prior knowledge about the labels. For example, in the next section we consider an exact active learning problem where nature is assumed to answer consistently with a known hypothesis class.

Making assumptions about nature changes the definition of "worst case cost" and "worst case value": the worst case cost (value) may be smaller (larger) if nature is restricted such that certain sequences of responses are no longer allowed. This changes the lower bound portion of the analysis (Lemma 2.1 and Lemma 2.4). Recall Lemma 2.1 showed that if there is an algorithm with worst case cost C^* then $GCC \leq C^*$. If we place restrictions on nature which decrease C^* then the lemma may no longer hold. In this chapter we identify sufficient conditions under which $GCC \leq C^*$ and therefore our analysis still holds.

These sufficient conditions will prove useful in future sections in which we discuss several active learning and interactive optimization problems which can be reduced to interactive submodular set cover. Many of these problems, such as the exact active learning problem in the next section, make assumptions about nature. When reducing these problems to interactive submodular set cover we are careful to ensure the sufficient conditions discussed in this chapter hold. By doing so we ensure that the resulting algorithm is approximately optimal under the assumptions about nature made by the original problem.

We consider restrictions parametrized by a set function $G(S)$ and a threshold κ.

Definition 2.1. *We say that nature is* (G, κ)-restricted *if it is always true that after asking any number of questions* $G(S_i) < \kappa$ *where* $S_i = \bigcup_{j \leq i} \{(q_j, r_j)\}$ *is the set of observed question-response pairs.*

In some sense G plays the role of a penalty function: if nature is (G, κ)-restricted then nature is assumed to never incur κ penalty. We have so far placed no restrictions on $G(S)$, so this class of restrictions is extremely general. We consider a particular sub-class of restrictions defined below.

Definition 2.2. *We say that* (G, κ) *is* adversarially uncoverable *if for any* $S \subseteq (Q \times R)$ *with* $G(S) < \kappa$ *there is an oracle* $T \in R^Q$ *such that for any* $\hat{Q} \subseteq Q$, $G(S \cup T(\hat{Q})) < \kappa$

Stated differently, if (G, κ) is adversarially uncoverable then for any starting set $S \subseteq Q \times R$ with $G(S) < \kappa$ an adversary can always respond to questions such that (G, κ) is never covered.

For (G, κ) adversarially uncoverable the assumption that nature is (G, κ)-restricted becomes a much simpler, easier to analyze assumption. To see this, consider the problem from the perspective of nature: say we are asked a sequence of questions and our goal is to ensure $G(S) < \kappa$. If (G, κ) is adversarially uncoverable then this game becomes easy. So long as $G(\emptyset) < \kappa$ and we myopically choose each r_i so that $G(S_{i-1} + (q_i, r_i)) < \kappa$ we are guaranteed to keep $G(S) < \kappa$.

If on the other hand $G(S)$ were an arbitrary set function this problem could be arbitrarily hard. In fact, if $G(S)$ is arbitrary, simply determining if it is possible for nature to be (G, κ) restricted requires exponential time. To see this note that it is possible for nature to be (G, κ) restricted if and only if there is any oracle T such that $G(T(Q)) < \kappa$. If $G(S)$ is an arbitrary set function than the value $G(T(Q))$ for one oracle T provides no information about the value $G(T'(Q))$ for another oracle T'. We are therefore forced to enumerate all $|R|^{|Q|}$ oracles to check if there is one which ensures nature is (G, κ) restricted.

From another perspective, say we are observing a sequence of question response pairs and it is our job to check whether or not nature is (G, κ)-restricted. If G is adversarially

uncoverable then to confirm that nature is *not* (G, κ)-restricted it is both necessary and sufficient to observe $G(S) \geq \kappa$. This is not the case for more general G. For more general G it may be that nature is locally (G, κ)-restricted (e.g. $G(S) < \kappa$) but that there is some sequence of questions which always forces nature to violate the assumption. In a sense, if (G, κ) is adversarially uncoverable then assuming nature is (G, κ)-restricted becomes a "local" assumption.

We show that if (1) (G, κ) is adversarially uncoverable and (2) $F(S) = F(Q \times R)$ if $G(S) \geq \kappa$ then our guarantees carry over to the setting where nature is (G, κ)-restricted. As we just discussed, assuming (G, κ) is adversarially uncoverable simplifies the restriction on nature. The second assumption additionally ensures that our objective function in some way incorporates our prior knowledge about nature: the penalty function G is encoded into F in the sense that F is covered when (G, κ) is covered.

We start with the analysis for interactive submodular set cover.

Lemma 2.6. *Assume nature is (G, κ)-restricted where (G, κ) is adversarially uncoverable and $F(S) = F(Q \times R)$ if $G(S) \geq \kappa$. If there is an algorithm which is correct on the interactive submodular set cover instance (Q, R, F, c) with finite worst case cost C^* then $GCC(Q, R, F, c) \leq C^*$*

Proof. Assume $GCC(Q, R, F, c) > C^*$. We reach a contradiction by showing nature can answer questions in such a way that (1) we do not violate the (G, κ)-restricted assumption and (2) any algorithm must incur more than C^* cost. From the definition of GCC there is an oracle T^* such that for any set of questions $\hat{Q}$ with $c(\hat{Q}) \leq C^*$, $F(T^*(\hat{Q})) < F(Q \times R)$. However, if nature were to use T^* to answer questions it could possibly violate the assumption that nature is (G, κ)-restricted.

As before let (q_i, r_i) denote the ith question-response pair, $S_i \triangleq \bigcup_{j \leq i} \{(q_i, r_i)\}$, and $Q_i \triangleq \bigcup_{j \leq i} \{q_i\}$. Consider the following strategy for answering question q_i. If $c(Q_{i-1} + q_i) \leq C^*$ then answer according to $T^*(q_i)$. Responding in this way guarantees that $F(S_i) < F(Q \times R)$ and therefore any algorithm incurs at least C^* cost before $F(S_i) = F(Q \times R)$. Responding in this way also ensures that $G(S_i) < \kappa$ because by assumption $F(S_i) = F(Q \times R)$ if $G(S_i) \geq \kappa$. Consider now the first question q_i with $c(Q_{i-1} + q_i) > C^*$. At this point $G(S_{i-1}) < \kappa$. From

the definition of adversarially uncoverable we know that there is an oracle $T \in R^Q$ such that for any $\hat{Q} \subseteq Q$, $G(S_i \cup T(\hat{Q})) < \kappa$. If we answer q_i and any remaining questions according to this T then we ensure $G(S_i) < \kappa$ no matter how many additional questions are asked. □

Combining Lemma 2.6 with Lemma 2.3 we have the following. Note that the upper bound portion of the analysis (Lemma 2.3) does not change because restricting nature can only help the greedy algorithm.

Theorem 2.3. *Assume nature is (G, κ)-restricted where (G, κ) is adversarially uncoverable and $F(S) = F(Q \times R)$ if $G(S) \geq \kappa$. If there is an algorithm which is correct on the interactive submodular set cover instance (Q, R, F, c) with worst case cost C^* then a $\frac{1}{(1-\epsilon)}$ worst case greedy algorithm incurs cost C with*

$$C \leq C^* \frac{1}{(1-\epsilon)} (1 + \ln \frac{F(Q \times R)}{\delta})$$

Note that Theorem 2.3 holds despite the fact that the worst-case greedy algorithm does not make use of the assumption nature is (G, κ)-restricted while the algorithms it is compared to may. In essence this theorem shows that these assumptions cannot help any other algorithm too much. This is a somewhat surprising result. The intuitive reason the result holds is that we are only restricting nature in a relatively simple way (because (G, κ) is adversarially uncoverable) and in a way that is compatible with the objective (because $F(S) = F(Q \times R)$ if $G(S) \geq \kappa$).

We now turn our attention to the maximization version of the problem.

Lemma 2.7. *Assume nature is (G, κ)-restricted where (G, κ) is adversarially uncoverable and $F(S) = F(Q \times R)$ if $G(S) \geq \kappa$. If there is a correct algorithm for interactive submodular maximization instance (Q, R, F, k) achieving worst case value α^* then*

$$GCC(Q, R, F_{\alpha^*}, c) \leq k$$

where $F_{\alpha^*}(S) = \min(F(S), \alpha^*)$ and $c(q) = 1$ for all $q \in Q$.

Proof. Again note that F_{α^*} is submodular, monotone, and non-negative (Proposition 1.6). Also note that if $F(S) = F(Q \times R)$ when $G(S) \geq \kappa$ then it is also the case that $F_{\alpha^*}(S) = F_{\alpha^*}(Q \times R)$ when $G(S) \geq \kappa$. Therefore the algorithm described implies an algorithm for the interactive submodular set cover problem (Q, R, F_{α^*}, c) with worst case cost k under the assumptions in Lemma 2.6. The Lemma then follows from Lemma 2.6. □

Combining Lemma 2.4 and Lemma 2.5 we have the following.

Theorem 2.4. *Assume nature is (G, κ)-restricted where (G, κ) is adversarially uncoverable and $F(S) = F(Q \times R)$ if $G(S) \geq \kappa$. If there is an algorithm for interactive submodular maximization instance (Q, R, F, k) achieving worst case value α^* then a $(1-\epsilon)$ approximate worst case greedy algorithm achieves value α with*

$$\alpha \geq (1 - (\frac{1}{e})^{1-\epsilon})\alpha^*$$

2.7 Connection to Exact Active Learning

One of the original motivations for this work was the apparent similarity between the greedy algorithm for submodular set cover and the greedy algorithm for exact active learning [Balcázar et al., 2007, Hanneke, 2006]. In particular, the standard greedy query learning algorithm approximates the optimal algorithm within a factor of $\ln |H|$ [Hanneke, 2006]. The standard greedy algorithm for submodular set cover gives a $\ln F(V) + 1$ approximation for integer valued objectives. It seems unlikely this similarity is a coincidence. We show that in fact interactive submodular set cover is a natural generalization of both problems and that the worst-case greedy algorithm matches the known approximation results (up to constant factors).

There are many alternative formulations of exact active learning (query learning). We use a general formulation similar to that of Hanneke [2006]. The goal of query learning is to identify a *target hypothesis* $h^* \in H$ (H is our *hyopthesis class*) using a minimal cost sequence of questions. We define questions to map hypotheses to sets of responses. For every $q \in Q$, $h \in H$, there is a known set $q(h) \subseteq R$. Intuitively $q(h)$ defines the set of

allowed responses to q if h were the target hypothesis. When we ask a question q we receive a response $r \in q(h^*)$. We can therefore safely conclude that $h^* \neq h$ if $r \notin q(h)$. The problem is formalized below.

Exact Active Learning

Given:

- Hypothesis class H containing unknown $h^* \in H$
- Query set Q, response set R with known non-empty $q(h) \subseteq R$ for $q \in Q$, $h \in H$
- Modular, positive query cost function c defined over Q

Protocol: At step i algorithm asks $q_i \in Q$, receives a response $r_i \in q_i(h^*)$ from nature

Goal: Identify h^* using minimal cost questions

As an example of this framework, say we are learning a binary classifier from examples. In this case we can set H to be a set of possible binary classifiers (e.g. decision trees of a certain depth). Q would be the set of examples available to the learner and R would be $\{0, 1\}$. $q(h)$ would then be $\{1\}$ for if h classifies the example q positively and $\{0\}$ if h classifies the example negatively. Finally, h^* is the (assumed unique) zero error classifier.

The setting we have defined is more general than alternative settings where it is assumed there is a single valid response for each question-hypothesis pair ($|q(h)| = 1$). For example, say that the learning algorithm can request a positively labeled example from a set of unlabeled examples. For this kind of query there may be multiple valid responses for a fixed labeling of the data set (a fixed hypothesis).

For a set of question-response pairs S define the version space $V(S)$ to be the subset of H consistent with S. More formally

$$V(S) \triangleq \{h \in H : \forall (q, r) \in S, \quad r \in q(h)\} \tag{2.1}$$

Exact active learning is about version space reduction. If we assume worst case choice of h^* then in order to identify h^* it is both necessary and sufficient to ensure $|V(S)| = 1$: if we have not ensured $|V(S)| = 1$ then there are multiple h for which we cannot conclude $h^* \neq h$. This view of active learning also leads to a connection with submodularity: version

space reduction is a submodular function.

Lemma 2.8. *$F(S) \triangleq |H \setminus V(S)|$ is monotone, submodular*

Proof. Monotonicty is clear from the definition. To show submodularity note that

$$F(S) = \sum_{h \in H} \max_{(q,r) \in S} f_h((q,r))$$

where $f_h((q,r)) = I(r \notin q(h))$ (I is the indicator function). Submodularity then follows from Propositions 1.5 and 1.7. □

Combining Lemma 2.8 with Theorem 2.3 gives an approximation result for exact active learning through the worst-case greedy algorithm. Note that in this problem we analyze worst case cost with respect to worst case choice of both h^* and the responses consistent with h^*.

Theorem 2.5. *Assume there is an algorithm for the active learning problem (Q, R, H, c) with worst case cost C^*. Applying the worst case greedy algorithm to $F(S) = \min(|H \setminus V(S)|, |H| - 1)$ identifies h^* using worst case cost C with*

$$C \leq (1 + \ln |H|)C^*$$

Proof. From Lemma 2.8 and Proposition 1.6 $F(S)$ is submodular, monotone, non-negative. Achieving $|V(S)| \leq 1$ is equivalent to satisfying $F(S) = F(Q \times R)$. Identifying h^* is equivalent to ensuring $|V(S)| = 1$.

Define $G(S) = |H \setminus V(S)|$. Assuming nature answers consistently with some $h^* \in H$ is equivalent to assuming nature is $(G, |H|)$-restricted. Moreover, assuming $|q(h)| \geq 1$ for every q and h ensures that $(G, |H|)$ is adversarially uncoverable. Also clearly $G(S) \geq |H|$ implies $F(S) = F(Q \times R)$. Therefore an active learning with worst case cost C^* on (Q, R, H, c) implies an algorithm for interactive submodular set cover problem (Q, R, F, c) which, under the assumptions in Theorem 2.3, also has worst case cost C^* . The result then follows from Theorem 2.3. □

The bound in this theorem is not new. Hanneke [2006] showed a similar result in this same kind of general setting with arbitrary queries and costs. However, it is interesting to see the result re-derived as part of a more general interactive submodular optimization problem.

2.8 Hardness

Any approximation algorithm asks the question: can we do better? We show that under reasonable complexity assumptions there is no better polynomial time algorithm. The result follows from the close relationship between interactive submodular set cover and set cover. In particular when $|R| = 1$ the problem becomes submodular set cover which generalizes set cover.

Lemma 2.9. *If for some $\epsilon > 0$ there is a polynomial time algorithm for interactive submodular set cover with worst case cost $(1 - \epsilon) \ln \frac{F(V)}{\delta}$ times that of the optimal policy then $NP \subset TIME(n^{O(\log \log n)})$.*

Proof. We can reduce a set cover problem to an interactive submodular set cover problem with $Q = V$, $|R| = 1$ and F equal to the set cover objective (1.4). Note that with $|R| = 1$ the problem becomes equivalent to submodular set cover. A $(1 - \epsilon) \ln \frac{F(V)}{\delta}$ approximation algorithm for interactive submodular set cover therefore implies a $(1-\epsilon) \ln n$ approximation algorithm for the corresponding set cover problem. The result then follows from Feige [1998] □

A similar result for the maximization version of the problem also follows directly from Feige [1998].

Corollary 2.2. *If for some $\epsilon > 0$ there is a polynomial time algorithm for interactive submodular maximization with worst case value $(1-1/e-\epsilon)$ times that of the optimal policy then $P = NP$.*

2.9 Adaptivity Gap

The algorithm we have proposed is an *adaptive* or interactive algorithm: it uses feedback from previous questions in order to pick the next question to ask. An alternative would be to

consider only non-adaptive algorithms. The *adaptivity gap* [Dean et al., 2004] for a problem characterizes how much worse the best non-adaptive method can perform as compared to the best adaptive method. For interactive submodular set cover we define the adaptivity gap for a problem instance (Q, R, F, c) to be the ratio between the worst case cost of the optimal non-adaptive strategy and the optimal adaptive strategy. With this definition, we can show that, in contrast to related problems [Asadpour et al., 2008] where the adaptivity gap is a constant, the adaptivity gap for interactive submodular set cover is quite large.

Theorem 2.6. *For any positive integer n there is an interactive submodular set cover problem (Q, R, F, c) where the adaptivity gap is at least $(2^n - 1)/n$*

Proof. The result follows directly from the connection between interactive submodular set cover and active learning and in particular any example of exact active learning giving an exponential speed up over passive learning. One classic example is learning a threshold on a line Dasgupta [2004]. Let $|H| = 2^n$. Define F to be the exact active learning objective (Theorem 2.5).

We define the query set such that we can satisfy $|V(S)| \leq 1$ through binary search. Let there be $2^n - 1$ queries with $q_i(h_j) = \{1\}$ if $i < j$ and $q_i(h_j) = \{0\}$ if $i \geq j$. Each q_i can be thought of as a point on a line with h_i the binary classifier which classifies all points as positive which are less than q_i. By asking question $q_{2^{n-1}}$ we can eliminate half of $H \setminus \{h^*\}$ from the version space. By applying this strategy recursively we satisfy $|V(S)| = 1$ in n queries. Any non-adaptive strategy on the other hand must perform all $2^n - 1$ queries in order to ensure $|V(S)| = 1$ for worst case choice of responses. □

Chapter 3

MULTIPLE OBJECTIVES

The interactive submodular optimization problems we have defined so far are simple in the sense that they take in minimal inputs and make minimal assumptions. In this chapter and the next we will show that several interesting more general problems can be reduced to interactive submodular set cover. We have already seen one example of this: Corollary 2.1 which introduces an explicit threshold α. In this chapter we examine a more complicated problem involving a set of objectives $\{F_h\}$ parametrized by a hypothesis class H. Perhaps surprisingly, this more complex problem reduces to the simple single objective problem. We also show how this more complex problem can be used for approximate active learning. In the proceeding chapter we then extend these problems to allow for noise. Figure 3.1 shows a map of these reductions. We also discuss how the multiple objective version of interactive submodular set cover can be applied to an interesting new class of applications we call *simultaneous learning and covering problems*.

3.1 Problem

Below we formally define the multiple objective version of interactive submodular set cover.

Interactive Submodular Set Cover (Multiple Objective Version)

Given:

- Hypothesis class H containing unknown $h^* \in H$
- Query set Q, response set R with known non-empty $q(h) \subseteq R$ for $q \in Q$, $h \in H$
- Submodular monotone $F_h : 2^{Q \times R} \to \mathbb{R}_{\geq 0}$ for every $h \in H$
- Modular, positive query cost function c defined over Q

Protocol: At step i algorithm asks $q_i \in Q$, receives a response $r_i \in q_i(h^*)$ from nature

Goal: Achieve $F_{h^*}(\bigcup_{j \leq i} \{(q_j, r_j)\}) = F_{h^*}(Q \times R)$ using minimal cost $\sum_{j \leq i} c(q_i)$

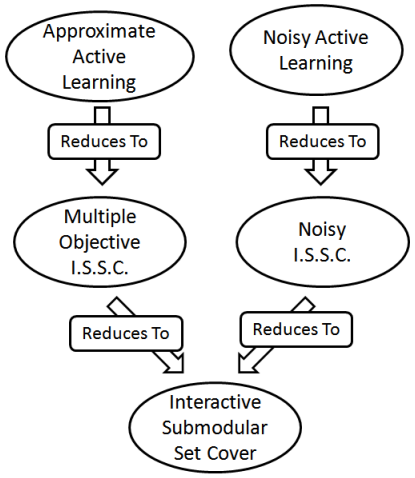

Figure 3.1: Reductions to interactive submodular set cover.

As compared to the simple version of interactive submodular set cover, this problem adds a hypothesis class H (as in exact active learning) and replaces the single objective F with a set of objectives $\{F_h\}$ for $h \in H$. The goal is no longer to satisfy $F(S) = F(Q \times R)$ but rather $F_{h^*}(S) = F_{h^*}(Q \times R)$ where h^* is the (initially unknown) target hypothesis. One way of viewing this problem is as a submodular set cover problem where the coverage constraint depends on the solution to an exact active learning problem.

The simple version of interactive submodular set cover is a special case of this problem. In particular the simple problem corresponds to the case where $|H| = 1$ and $q(h) = R$. Interestingly, we will give an algorithm for the multiple objective problem by reducing it to the simple problem. In some sense therefore the two problems are equivalent.

We note that in order to solve this problem it is not strictly necessary to identify h^*. For example, it would suffice to instead ensure $F_h(S) = F_h(Q \times R)$ for every $h \in H$. We will in fact show that in general any algorithm which always identifies h^* has poor approximation ratio. However, we also show that any algorithm which does not incorporate feedback also has poor approximation ratio. Therefore it is necessary to balance between learning about h^* and achieving coverage. This balance is similar to the exploration-exploitation trade off

in reinforcement learning.

3.2 Learning and Covering Problems

The multiple objective problem suggests an interesting new class of applications which we call simultaneous learning and covering problems. The class of problems is best described through an example.

Say we are performing viral marketing over a social network. As discussed in Section 1.3.4, this can be posed as a submodular maximization or submodular set cover problem. In particular we can define a submodular function $F(S)$ which defines the influence achieved by inserting ads at nodes S. Many reasonable measures of influence turn out to be submodular [Kempe et al., 2005]. Selecting a minimal set of nodes to achieve some target level of influence is then a submodular set cover problem while selecting the best k nodes to advertise to is (cardinality constrained) submodular maximization.

Through interactive submodular set cover we can define versions of this problem which incorporate learning and feedback. For example, say we know that the particular product we are advertising is only relevant to some subset of the network. Additionally assume that we *do not* initially know which members of the network fall into this subset but that we receive feedback from our actions (e.g. through the presence or absence of ad clicks) through which we can learn about the target group. We would like to simultaneously learn about the target group and perform actions to achieve high influence in the group.

In this example application, H would model our assumptions about the target group. If we know that members of the target group form a large dense cluster in the social network, we can set H to be the set of all large dense clusters in the network. Q and R would be advertising actions and feedback respectively. Finally $F_h(S)$ would be an objective measuring the influence achieved by the actions in S *assuming* h *is the target group.*

Other applications that share this structure include simultaneously diagnosing and treating a patient's illness and simultaneously learning a user's taste in movies and making a small set of good recommendations.

3.3 Greedy Algorithm and Analysis

Our approach to the multiple objective version of interactive submodular set cover is to reduce it to a simpler single objective problem. Then we can apply the analysis of the previous section. The particular objective we use is

$$\bar{F}(S) \triangleq \sum_{h \in V(S)} F_h(S) + \sum_{h \notin V(S)} F_h(Q \times R)$$

Here $V(S)$ is the version space. Intuitively the first sum encodes our preference for increasing $F_h(S)$ for h currently in the version space. The second sum encodes our preference for decreasing the size of the version space.

This objective is inspired by an objective used by Krause et al. [2008a] for a non-interactive worst case optimization problem. Krause et al. [2008a] convert a collection of constraints of the form $F_i(S) = F_i(V)$ to a single constraint $\bar{F}(S) = \bar{F}(V)$ by summing over the individual objectives. Krause et al. [2008a] use this to give a bi-criteria approximation algorithm for robust submodular maximization:

$$\max_{S:|S| \leq k} \min_i F_i(S) \tag{3.1}$$

Note this problem is essentially the non-interactive, maximization version of our problem.

It is not hard to show that covering $\bar{F}(S)$ is sufficient and necessary.

Lemma 3.1. *For an algorithm to ensure $F_{h^*}(S) = F_{h^*}(Q \times R)$ for any h^* it is necessary and sufficient to ensure $\bar{F}(S) = \bar{F}(Q \times R)$*

Proof. Sufficiency is clear. If $\bar{F}(S) < \bar{F}(Q \times R)$ then there is some $h \in V(S)$ with $F_h(S) < F_h(Q \times R)$. Since $h \in V(S)$ we cannot concluded $h^* \neq h$ and therefore we cannot conclude $F_{h^*}(S) = F_{h^*}(Q \times R)$ and covering $\bar{F}$ is necessary. □

Critically, $\bar{F}(S)$ retains submodularity and monotonicity.

Lemma 3.2. *$\bar{F}(S)$ is submodular, monotone, non-negative when $F_h(S)$ is submodular, monotone, non-negative.*

Proof. From Proposition 1.7 it suffices to show that for any h, $F(S) = F_h(S)I(h \in V(S)) + F_h(Q \times R)I(h \notin V(S)$ is submodular and monotone. To show that it is monotone we show that for any S, (q, r), $F(S + (q, r)) - F(S) \geq 0$. If $h \notin V(S)$ then

$$F(S + (q, r)) - F(S) = 0$$

If $h \in V(S)$, $r \notin q(h)$ then from the monotonicity of F_h

$$F(S + (q, r)) - F(S) = F_h(Q \times R) - F_h(S) \geq 0$$

If $h \in V(S)$, $r \in q(h)$ then from the monotonicity of F_h

$$F(S + (q, r)) - F(S) = F_h(S + (q, r)) - F_h(S) \geq 0$$

To show that it is submodular we show that for any A, B with $A \subseteq B$ and any (q, r), $F(A + (q, r)) - F(A) \geq F(B + (q, r)) - F(B)$. If $h \notin V(B)$ then from monotonicity of F

$$F(A + (q, r)) - F(A) \geq 0 = F(B + (q, r)) - F(B)$$

If $h \in V(B)$, $r \notin q(h)$ then from the monotonicity of F_h

$$\begin{aligned} F(A + (q, r)) - F(A) &= F_h(Q \times R) - F_h(A) \\ &\geq F_h(Q \times R) - F_h(B) \\ &= F(B + (q, r)) - F(B) \end{aligned}$$

If $h \in V(B)$, $r \in q(h)$ then from the submodularity of F_h

$$\begin{aligned} F(A + (q, r)) - F(A) &= F_h(A + (q, r)) - F_h(A) \\ &\geq F_h(B + (q, r)) - F_h(B) \\ &= F(B + (q, r)) - F(B) \end{aligned}$$

□

We now show that applying the worst-case greedy algorithm to $\bar{F}(S)$ is approximately optimal. An instance of the multiple objective problem is specified by $(Q, R, H, \{F_h\}, c)$.

Theorem 3.1. *Assume $F_h(S)$ is integer valued for every h. If there is an algorithm for the multiple objective interactive submodular set cover problem $(Q, R, H, \{F_h\}, c)$ with worst case cost C^* then applying the worst case greedy algorithm to $\bar{F}(S)$ incurs cost C*

$$C \leq C^*(1 + \ln(\frac{\sum_{h \in H} F_h(Q \times R)}{\delta}))$$

where δ is the smallest non-zero gain of any of $F_h(S)$.

Proof. By 3.1 it is necessary and sufficient for an algorithm for $(Q, R, H, \{F_h\}, c)$ to cover $\bar{F}(S)$. Assuming nature answers consistently with some $h^* \in H$ then nature is equivalent to assuming nature is $(G, |H|)$-restricted with $G \triangleq |H \setminus V(S)|$. Again note that assuming $|q(h)| \geq 1$ for every q and h ensures that $(G, |H|)$ is adversarially uncoverable and $G(S) \geq |H|$ implies $\bar{F}(S) = F(Q \times R)$. Therefore an algorithm with worst case cost C^* on $(Q, R, H, \{F_h\}, c)$ implies an algorithm for the interactive submodular set cover problem $(Q, R, \bar{F}, c)$ which, under the assumptions in Theorem 2.3, also has worst case cost C^* . Then the result follows from Theorem 2.3. □

3.4 Connection to Approximate Active Learning

In Section 2.7 we showed that interactive submodular set cover is closely related to exact active learning. Here we show that the multiple objective problem is closely related to approximate active learning. We assume as input to the problem some distance measure over hypotheses $d(h, h')$.

Definition 3.1. *Define a distance measure $d(h, h')$ over H to be a function satisfying*

$$d(a, c) \leq d(a, b) + d(b, c)$$

for any a, b, c in H.

Note that more precisely this is a pseudo-metric: we do not require $d(h, h') = 0$ iff $h = h'$. We define the approximate active learning problem below to be the problem of identifying some h with $d(h, h^*) \leq \epsilon$.

Approximate Active Learning

Given:

- Hypothesis class H contaning unknown $h^* \in H$
- Query set Q, response set R with known $q(h) \subseteq R$ for $q \in Q$, $h \in H$
- Modular, positive query cost function c defined over Q
- Distance measure $d(h, h')$ over H (Definition 3.1)
- Accuracy ϵ

Protocol: At step i algorithm asks $q_i \in Q$, receives a response $r_i \in q_i(h^*)$ from nature

Goal: Identify $h \in H$ with $d(h, h^*) \leq \epsilon$ using minimal cost questions

To make the abstract problem more clear, consider learning a binary classifier. In this case Q is a set of unlabeled data and $q(h) = \{0\}$ or $q(h) = \{1\}$ is the label assigned to q by classifier h. We can define d to be

$$d(h, h') = \sum_{q \in Q} I(q(h) \neq q(h'))$$

$d(h, h')$ is the hamming distance and can be thought of as the number of data points on which h and h' disagree. $d(h, h^*)$ is the number of mistakes made by h.

Define $F_h(S)$ as follows

$$F_h(S) \triangleq \sum_{h' : d(h,h') > \epsilon} I(h' \notin V(S)) \tag{3.2}$$

Lemma 3.3. *(3.2) is submodular, monotone, non-decreasing*

Proof. From Proposition 1.7 it suffices to show that for any h, $F(S) = I(h \notin V(S))$ is submodular, monotone. In fact this is a slight variation of Lemma 2.8.

$$F(S) = \max_{(q,r) \in S} f_h((q, r))$$

where $f_h((q, r)) = I(r \notin q(h))$. Submodularity then follows from Proposition 1.5. □

If we satisfy $F_{h^*}(S) = F_{h^*}(Q \times R)$ then we are guaranteed that any hypothesis in $h \in V(S)$ has $d(h, h^*) \leq \epsilon$. Therefore covering F_{h^*} is sufficient for approximate active learning.

Corollary 3.1. *To solve the approximate active learning problem* (Q, R, H, ϵ, c) *it is sufficient to solve the multiple objective interactive submodular set cover problem* $(Q, R, H, \{F_h\}, c)$ *with (3.2)*

Proof. After solving the interactive submodular set cover problem any $h \in V(S)$ is a valid solution to the approximate active learning problem. □

However, ensuring that *every* $h \in V(S)$ is close to h^* is potentially a harder problem than identifying a *single* $h \in H$ close to h^*. In other words it may not be *necessary* to cover F_{h^*} in order to solve the approximate active learning problem. We in fact show that satisfying $F_{h^*}(S) = F_{h^*}(Q \times R)$ is equivalent to a different problem: finding a small diameter ball containing h^*.

Lemma 3.4. *Identifying a set* $\hat{H} \subseteq H$ *such that* $\max_{h \in \hat{H}, h' \in \hat{H}} d(h, h') \leq \epsilon$ *and* $h^* \in \hat{H}$ *is equivalent to solving the multiple objective interactive submodular set cover problem* $(Q, R, H, \{F_h\}, c)$ *with (3.2)*

Proof. From Lemma 3.1 an algorithm for $(Q, R, H, \{F_h\}, c)$ must ensure $F_h(S) = F_h(Q \times R)$ for every $h \in V(S)$. In this case $\max_{h \in V(S), h' \in V(S)} d(h, h') \leq \epsilon$ and $h^* \in V(S)$, so solving $(Q, R, H, \{F_h\}, c)$ is sufficient.

We now show that solving $(Q, R, H, \{F_h\}, c)$ is necessary. Consider any algorithm which identifies an $\hat{H}$ as described. We must have $V(S) \subseteq \hat{H}$ where S is the set of questions asked by the algorithm. Otherwise some valid choice of h^* makes $h^* \notin \hat{H}$. Taking a subset of $\hat{H}$ only decreases the diameter, so $\max_{h \in V(S), h' \in V(S)} d(h, h') \leq \epsilon$. Therefore $F_h(S) = F_h(Q \times R)$ for every $h \in V(S)$ and the algorithm also solves $(Q, R, H, \{F_h\}, c)$. □

This Lemma is interesting on its own: identifying a low diameter ball containing h^* is a reasonable approximate active learning objective. However, we also show this Lemma can be used to derive a bi-criteria approximation for approximate active learning. This theorem makes use of the close relationship between between identifying low diameter balls and low radius balls.

Theorem 3.2. *Assume there is an algorithm for the approximate active learning problem* (Q, R, H, ϵ, c) *with worst case cost* C^**. Applying the worst-case greedy algorithm to* $\bar{F}(S)$

with (3.2) identifies an h with $d(h, h^*) \leq 2\epsilon$ using cost

$$C \leq (1 + 2 \ln |H|)C^*$$

Proof. If there is such an active learning algorithm then there is also an algorithm for identifying a diameter 2ϵ ball containing h^* using worst case cost C^*: we simply run the active learning algorithm to identify an h with $d(h, h^*) \leq \epsilon$, then take a ball of radius ϵ entered at h. From Lemma 3.4 this is also an algorithm for the multiple objective interactive submodular set cover problem $(Q, R, H, \{F_h\}, c)$. The result then follows from Theorem 3.1 □

This is a bicriteria approximation because we are comparing the cost our algorithm incurs identifying an h with $d(h, h^*) \leq 2\epsilon$ to that of the optimal algorithm for identifying an h with $d(h, h^*) \leq \epsilon$ (a harder problem). This result is similar to a result of Balcázar et al. [2007] for approximate query learning. It also similar in spirit to the approximate active learning results of Dasgupta [2006] and Hanneke [2006] although these results consider stochastic settings. When F_h is chosen to be (3.2), the composite objective $\bar{F}$ becomes the same as the edge cutting objective first proposed by Dasgupta [2006] and later used by Golovin et al. [2010b].

3.5 Connection to Adaptive Submodularity

In concurrent work, Golovin and Krause [2010] give results for a different but related class of problems which also involve interactive (i.e. sequential, adaptive) optimization of submodular functions parametrized by a hypothesis class. What Golovin and Krause call realizations correspond to hypotheses in our work while items and states correspond to queries and responses respectively. Golovin and Krause consider both average-case and worst-case settings and both maximization and submodular set cover type problems. In contrast, we only consider worst-case cost and, except for the simple, one objective maximization problem, we only consider submodular set cover problems. In this sense our results are less general.

In other ways our results are more general. The main greedy approximation guarantees shown by Golovin and Krause require that the problem is *adaptive submodular*; adaptive submodularity depends not only on the objective but also on the set of hypotheses and a probability distribution over these hypotheses. More formally, Golovin and Krause assume h^* is chosen at random according to a known probability distribution $P(h^*)$. They also assume that hypotheses map questions to single responses $|q(h)| = 1$. Under these assumptions we can define the conditional probability that $h^* = h$ given a set of question response pairs S

$$P(h^* = h|S) = I(h \in V(S))\frac{P(h^* = h)}{P(h^* \in V(S))}$$

and define the expected gain of a question to be

$$\Delta(q|S) \triangleq \sum_{h \in H} P(h^* = h|S)(F_h(S \cup q(h)) - F_h(S))$$

Note this is only defined if $P(h^* \in V(S)) > 0$. Adaptive submodularity requires that for any q $\Delta(q|S)$ is non-increasing with respect to S. Similarly, adaptive monotonicity requires that $\Delta(q|S)$ is non-negative for any q, S.

In contrast, the multiple objective problem in this section requires only that for any fixed hypothesis the objective is submodular. Golovin and Krause call this pointwise submodularity. Pointwise submodularity does not in general imply adaptive submodularity. In fact, there are certain H and probability distributions over h^* for which even point-wise *modular* objectives are not adaptive submodular [Golovin and Krause, 2010]. Point-wise submodularity is in some ways easier to work with than adaptive submodularity: to show an objective is point-wise submodular we only need to show submodularity for every hypothesis, and to do this we can use all of the standard tools from the analysis of submodular functions. Showing a problem is adaptive submodular requires a more subtle argument concerning the expected reward of algorithms under different conditional probability distributions.

Interestingly, adaptive submodularity also doesn't imply point-wise submodularity. This is the case if the individual objectives F_h are not submodular but H and $P(h^*)$ are such that the objectives are submodular on average over any version space. Therefore it would seem that neither point-wise submodularity nor adaptive submodularity is strictly more general

than the other. However, we show that for worst-case cost the adaptive submodular version of submodular set cover can be reduced to a interactive submodular set cover problem over a single objective. This suggests that for worst-case submodular set cover type problems our results are more general. The key result is the following lemma.

Lemma 3.5. *If the set of objectives* $\{F_h : h \in H\}$ *is adaptive monotone and adaptive submodular under the distribution* P *then*

$$\bar{F}(S) = \sum_{h \in V(S)} P(h^* = h) F_h(S) + \sum_{h \notin V(S)} P(h^* = h) F_h(Q \times R)$$

is monotone and submodular.

Proof. Assuming that $|q(h)| = 1$, the gain of $v = (q, r)$ with respect to $A \subseteq ((Q \times R) \setminus v)$ can be written

$$\begin{aligned}
\bar{F}(A + v) - \bar{F}(A) &= \sum_{h \in V(A)} P(h^* = h)(F_h(A \cup q(h)) - F_h(A)) \\
&\quad + \sum_{h \in (V(A) \setminus V(A+v))} P(h^* = h)(F_h(Q \times R) - F_h(A \cup q(h))) \\
&= P(h^* \in V(A))\Delta(q|A) \\
&\quad + \sum_{h \in (V(A) \setminus V(A+v))} P(h^* = h)(F_h(Q \times R) - F_h(A \cup q(h)))
\end{aligned}$$

If $\Delta(q|A) \geq 0$ (adaptive monotonicity) then this is non-negative and therefore F is monotone. Similarly for $A \subseteq B \subseteq ((Q \times R) \setminus v)$

$$\begin{aligned}
\bar{F}(B + v) - \bar{F}(B) &= P(h^* \in V(B))\Delta(q|B) \\
&\quad + \sum_{h \in (V(B) \setminus V(B+v))} P(h^* = h)(F_h(Q \times R) - F_h(B \cup q(h)))
\end{aligned}$$

and this is term by term less than the gain with respect to A, implying submodularity. □

Note that this objective is similar to the $\bar{F}$ function used to reduce the multiple objective version of interactive submodular set cover to a single objective problem. The only difference is that here the objectives are reweighted by the distribution $P(h^*)$. It is again the case that $\bar{F}$ is saturated if and only if $F_{h^*}(S) = F_{h^*}(Q \times R)$ for worst case choice of h^*. Having

reduced the problem to a single objective, the rest of the reduction follows Theorem 3.1. The minimum non-zero gain of the composite objective is $\delta(\min_{h \in H} P(h^* = h))$ where δ is the minimum non-zero gain of any F_h. Our worst-case cost approximation guarantee therefore matches the result of Golovin and Krause [2010].

For problems that are pointwise modular but not adaptive submodular, Golovin and Krause show a hardness of approximation lower bound of $|Q|^{1-\epsilon}$; this does not contradict our results as their proof is for average-case cost and uses a hypothesis class with $|H| = 2^{|Q|}$.

There are other smaller differences between our problem settings. For example, we let queries map hypotheses to *sets* of valid responses (in general $|q(h)| > 1$) while as previously mentioned Golovin and Krause define realizations as maps from items to *single* states. We finally note that the proof techniques we use are quite different. Our proofs draw heavily from worst case analyses of query learning while Golovin and Krause's analyses are more direct extensions of the classic (non-interactive) results.

Some other previous work has also considered interactive versions of covering problems in an average-case model [Asadpour et al., 2008, Goemans and Vondrák, 2006]. The work of Asadpour et al. [2008] is perhaps most similar and considers a submodular function maximization problem over independent random variables which are sequentially queried. The setting considered by Golovin and Krause [2010] strictly generalizes this setting.

3.6 Negative Results for Simpler Approaches

Our proposed algorithm is not the most obvious algorithm for the multiple objective problem. It simultaneously learns about h^* and attempts to cover $F_{h^*}(S)$ using a combined objective function $\bar{F}$. In this section we show that a number of simpler approaches have poor approximation ratios. Together these results show that in order to solve the multiple objective problem optimally it is necessary to consider both the learning and the covering aspects of the problem together.

3.6.1 Naïve Greedy

A simpler, more direct extension of the single objective algorithm is to choose a question which maximizes the worst case gain for worst case choice of h^*. In other words, at time

step i we pick the question q which maximizes

$$\min_{h \in V(S)} \min_{r \in q(h)} (F_h(S_{i-1} + (q, r)) - F_h(S_{i-1}))/c(q) \tag{3.3}$$

We call this the naïve greedy algorithm. This strategy is in contrast to our proposed approach which uses a more complex composite objective function. We show that the naïve greedy algorithm has a poor approximation ratio. This result is basically the same as the result of Krause et al. [2008a] for the non-interactive multiple objective maximization problem $\max_{S:|S| \leq k} \min_i F_i(S)$.

Theorem 3.3. *There is a positive constant k such that for any positive integer α there is a problem $(Q, R, H, \{F_h\}, c)$ where*

- *$F_h(S)$ is integer valued for every h*
- *$\sum_{h \in H} F_h(Q \times R) \leq k\alpha$*
- *The naïve greedy algorithm gives worst case cost C with $C \geq \alpha C^*$ where C^* is the optimal worst case cost*

Proof. Let $|H| = 2$, $|R| = 1$, $|Q| = \alpha + 2$. Note that with $|R| = 1$ responses provide no information about h^* so the problem is non-interactive. Define monotone objectives $\hat{F}_{h_1}$ and $\hat{F}_{h_2}$ with

$$\hat{F}_{h_1}(q_1) \triangleq \alpha \qquad \hat{F}_{h_1}(q_2) \triangleq 0$$
$$\hat{F}_{h_2}(q_1) \triangleq 0 \qquad \hat{F}_{h_2}(q_2) \triangleq \alpha$$

and, for all h and all q_i with $i > 2$, $\hat{F}_h(q_i) \triangleq 1$. Define $F_h(S)$ in terms of $\hat{F}_h(S)$ to be $F_h(S) \triangleq \min(\hat{F}_h(S), \alpha)$. For this problem $\sum_{h \in H} F_h(Q \times R) = 2\alpha$

The optimal strategy asks q_1 and q_2 (since h^* is unknown we must ask both). However, the gain according to (3.3) of q_1 or q_2 is zero while the gain of q_i for $i > 2$ is $1/c(q_i)$. The naïve greedy algorithm will then always ask every q_i for $i > 2$ before of asking q_1 and q_2. $\square$

Note that in this proof we only need uniform cost functions c to show the result. In fact the same basic proof also shows that for non uniform cost functions the cost of the naïve greedy algorithm can be arbitrarily worse than the optimal strategy, even when $\sum_{h \in H} F_h(Q \times R)$ is a constant.

3.6.2 Exact Learning Strategies

Another simple strategy is to first exactly identify h^* (for example using one of the algorithms for exact active learning which we have discussed) and then solve the single objective problem over F_{h^*}. If we assume that $|q(h)| = 1$ for every q and h then this single objective problem is a standard submodular set cover problem. We call this strategy which first identifies h^* then covers F_{h^*} the *learn then cover strategy*. This strategy is in contrast to our proposed approach which simultaneously learns about h^* and attempts to cover $F_{h^*}(S)$. We show that the learn then cover strategy and in fact any strategy which exactly identifies h^* in general has a poor approximation ratio.

Theorem 3.4. *There is a positive constant k such that for any positive integer α there is a problem $(Q, R, H, \{F_h\}, c)$ where*

- *$F_h(S)$ is integer valued for every h*
- *$\sum_{h \in H} F_h(Q \times R) \leq k\alpha$*
- *Any strategy that exactly identifies h^* has worst case cost C with $C \geq \alpha C^*$ where C^* is the optimal worst case cost*

Proof. To show the result we construct a problem for which identifying h^* is hard but satisfying $F_{h^*}(S) = F_{h^*}(Q \times R)$ is easy. Let $|H| = \alpha + 1$, $|Q| = |H| + 1$, $R = \{0, 1\}$ For $i \in 1...|H|$ let $q_i(h_j) = \{1\}$ if $i = j$ and $q_i(h_j) = \{0\}$ if $i \neq j$. For $i = |H| + 1$ let $q_i(h_j) = \{0\}$ for all j. In the worst case we need to ask α questions in order to identify h^*. However, if we define $F_h(S) \triangleq 1$ if S includes $q_{|H|+1}$ and $F_h(S) \triangleq 0$ otherwise the interactive submodular set cover problem is easy. To satisfy $F_{h^*}(S) = F_{h^*}(Q \times R)$ we simply need to ask question $q_{|H|+1}$. Note that $\sum_{h \in H} F_h(Q \times R) = \alpha + 1$. □

Note that again this proof only uses uniform cost functions c, and for non uniform cost functions the cost of the learn then cover strategy can be arbitrarily bad.

3.6.3 Cover All Strategy

An important distinction between this problem and exact active learning is that in order to satisfy $F_{h^*}(S) = F_{h^*}(Q \times R)$ it is *not* always necessary to identify h^*. For example, one simple strategy is to reduce the problem to a non-interactive problem: we can ignore entirely the responses we receive and simply ask a set of questions such that for any choice of h^* and any choice of responses consistent with h^* $F_{h^*}(S)$ is covered. We call this the *cover all strategy*. The cover all strategy is a non-adaptive approach. Therefore Theorem 2.6 shows this strategy has a poor approximation ratio.

An interesting aside, if we assume that $|q(h)| = 1$ for every q and h, it is possible to efficiently approximate the optimal non-adaptive policy. If we assume $|q(h)| = 1$ *and* $h = h^*$, $F_h(S)$ is completely determined by the set of questions we ask $\hat{Q}$. Use $\hat{F}_h(\hat{Q})$ to refer to this function. The goal of a non adaptive strategy is to find a minimal cost set $\hat{Q} \subseteq Q$ such that $\hat{F}_h(\hat{Q}) = \hat{F}_h(Q)$ for every h. We can now use the trick from Krause et al. [2008a] to convert the problem into a single objective problem. Define $\bar{F}(\hat{Q}) = \sum_h \hat{F}_h(\hat{Q})$. Clearly $\hat{F}_h(\hat{Q}) = \hat{F}_h(Q)$ for every h iff $\bar{F}(\hat{Q}) = \bar{F}(Q)$.

3.7 Maximization Version

We can also define a maximization version of the multiple objective problem where the goal is to maximize $F_{h^*}(S_k)$.

Interactive Submodular Maximization (Multiple Objective Version)

Given:

- Hypothesis class H contaning unknown $h^* \in H$
- Query set Q, response set R with known non-empty $q(h) \subseteq R$ for $q \in Q$, $h \in H$
- Submodular monotone, non-negative objectives $F_h(S)$ for every $h \in H$
- Positive integer k

Protocol: At step i algorithm asks $q_i \in Q$, receives a response $r_i \in q_i(h^*)$ from nature
Goal: Maximize $F_{h^*}(\bigcup_{i \leq k}\{(q_i, r_i)\})$

However, in contrast to the one objective case, there is unlikely to be an efficient approximation algorithm for this problem. This theorem follows directly from a result of Krause et al. [2008a] for non-interactive worst-case submodular maximization.

Theorem 3.5. *Unless $P = NP$ there is no polynomial time algorithm for the multiple objective version of interactive submodular maximization with approximately optimal worst-case value.*

Proof. In the special case where $|R| = 1$ $F_h(S)$ is completely determined by the set of questions we ask $\hat{Q}$ and can be re-written as a submodular function over questions. Use $\hat{F}_h(\hat{Q})$ to refer to this function. The goal of the maximization problem is then to find the set $\hat{Q} \subseteq Q$ with $|\hat{Q}| \leq k$ that maximizes $\hat{F}_{h^*}(\hat{Q})$. Since responses reveal no information about h^*, the worst-case value of an algorithm is $\min_{h \in H} \hat{F}_h(\hat{Q})$. In this case our problem is exactly the robust submodular maximization problem 3.1 of Krause et al. [2008a]. The result then follows directly from Krause et al. [2008a]. □

It may be possible to give a bicriteria approximation algorithm for this problem similar to the algorithm of Krause et al. [2008a] for the non-interactive problem. The basic approach is to guess the worst-case value of the optimal strategy α^*, then solve the multiple objective interactive submodular set cover problem for the constraint $F_{h^*}(S) \geq \alpha^*$. Recall that it is straightforward to convert constraints of this form to constraints of the form $F_{h^*}(S) = F_{h^*}(Q \times R)$ (Section 2.5). If we ensure $F_{h^*}(S) \geq \alpha^*$ in approximately k questions than we know our guess for α^* is approximately correct. An added difficulty in the interactive setting is that the cost incurred by an algorithm is the total number of questions asked and once a question is asked it cannot be undone. Because of this the way in which we search for α^* is crucial. It would be a bad idea to perform binary search for α^* as Krause et al. do in the non-interactive setting. If we started with an overestimate then we could

Table 3.1: Average number of queries to find a dominating set

Data Set / Hypothesis Class	Learning and Covering	Learn then Cover	Cover All
Enron / Clusters	**156.64**	161.81	3091.00
Physics / Clusters	**175.97**	**177.88**	3340.00
Physics Theory / Clusters	**172.38**	**175.12**	3170.00
Epinions / Clusters	**774.81**	779.23	15777.00
Slashdot / Clusters	**709.30**	715.39	15383.00
Enron / Noisy Clusters	**179.00**	231.03	3091.00
Physics / Noisy Clusters	**186.13**	225.02	3340.00
Physics Theory / Noisy Clusters	**160.62**	201.24	3170.00
Epinions / Noisy Clusters	**788.52**	**788.06**	15777.00
Slashdot / Noisy Clusters	**804.87**	**804.86**	15383.00

incur much more than k cost in the first step of the search. A better idea is to start with a underestimate of α^* and then repeatedly double the estimate until incurring about k cost.

3.8 Application to Viral Marketing

We tested our proposed algorithm for the multiple objective version of interactive set cover on a viral marketing inspired problem. The specific problem we consider is a simple, interactive version of dominating set. Recall that a dominating set in a graph is a set of vertices such that every vertex is either in the set or has a neighbor in the set. In the interactive version, we are given a graph and H is a set of possibly overlapping clusters of nodes. The goal is to find a small set of nodes which forms a dominating set of an initially unknown target group $h^* \in H$. After selecting each node, we receive feedback indicating if the selected node is in the target group.

More formally, we have a graph $G = (V, E)$. For each node $v \in V$ there is a query $q \in Q$ corresponding to selecting that node. For each $h \in H$ there is a set $V_h \subseteq V$ (H corresponds to a set of overlapping subsets of V). The set of responses $q(h)$ is $\{1\}$ if the

node corresponding to q is in V_h and $\{0\}$ otherwise. The objective is

$$F_h(S) \triangleq \sum_{v \in V_h} I\Big(v \in V_S \text{ or } \exists s \in V_S : (s, v) \in E\Big)$$

where V_S is the set of nodes corresponding to the queries in S. With this objective $F_{h^*}(S) = |V_{h^*}|$ iff we have selected a dominating set for V_{h^*}.

Lemma 3.6. *$F_h(S) \triangleq \sum_{v \in V_h} I\Big(v \in V_S$ or $\exists s \in V_S : (s, v) \in E\Big)$ is submodular and monotone non-decreasing.*

Proof. We can write $F_h(S)$ as $F_h(S) = \sum_{v \in V_h} \max_{(q,r) \in S} f_v((q, r))$ where $f_v((q, r)) = 1$ if the action q covers v and $f_v((q, r)) = 0$ otherwise. The result then follows from Propositions 1.5 and 1.7. □

This problem can be thought of as a viral marketing problem where the graph is a social network and selecting a node corresponds to sending someone an ad. After sending someone an ad we receive feedback indicating if that person was in the target group. Our goal is to send ads such that everyone in the target group either receives an ad or has a friend who receives an ad.

Our proposed method (Simultaneous Learning and Covering) simultaneously learns about the target group h^* and finds a dominating set for it. We compare to two baselines: a method which first exactly identifies h^* and then finds a dominating set for the target group (Learn then Cover) and a method which simply ignores feedback and finds a dominating set for the union of all clusters (Cover All). Note that our theoretical results (Section 3.6) show these methods do not have strong theoretical guarantees. However, we might hope however that for reasonable real world problems they perform well. We use real world network data sets with simple synthetic hypothesis classes designed to illustrate differences between the methods. The networks are from Jure Leskovec's collection of datasets available at `http://snap.stanford.edu/data/index.html`. We convert all the graphs into undirected graphs and remove self edges.

Table 3.1 shows our results. Each reported result is the average number of queries over 100 trials. Bolded results are the best methods for each setting with multiple results bolded

when differences are not statistically significant (within $p = .01$ with a paired t-test). In the first set of results (Clusters), we create H by using the METIS [Karypis and Kumar, 1999] graph partition package 4 separate times partitioning the graph into 10, 20, 30, and 40 clusters. H is the combined set of 100 clusters, and these clusters overlap since they are taken from 4 separate partitions of the graph. The target h^* is chosen at random from H. With this hypothesis class, we've found that there is very little difference between the Simultaneous Learning and Covering and the Learn then Cover methods. The Cover All method performs significantly worse because without the benefit of feedback it must find a dominating set of the entire graph.

In the second set of results, we use a hypothesis class designed to make learning difficult (Noisy Clusters). We start with H generated as before. We then add to H 100 additional hypotheses which are each very similar to h^*. Each of these hypotheses consists of the target group h^* with a random member removed. H is then the combined set of the 100 original hypotheses and these 100 variations of h^*. For this hypothesis class, Learn then Cover performs significantly worse than our Simultaneous Learning and Covering method on 3 of the 5 data sets. Learn then Cover exactly identifies h^*, which is difficult because of the many hypotheses similar to h^*. Our method learns about h^* but only to the extent that it is helpful for finding a small dominating set. On the other two data sets Learn then Cover and Simultaneous Learning and Covering are almost identical. These are larger data sets, and we've found that when the covering problem requires many more queries than the learning problem, our method is nearly identical to Learn then Cover.

It is also possible to design hypothesis classes for which Cover All outperforms Learn then Cover: we found this is the case when the learning problem is difficult but the subgraph corresponding to the union of all clusters in H is small. In all cases, however, our approach does about as good or better than the best of these two baseline methods. Although we use real world graph data, the hypothesis classes and target hypotheses we use are very simple and synthetic, and as such these experiments are primarily meant to provide reasonable examples to illustrate the theoretical results.

Chapter 4

NOISE AND IMPLICIT HYPOTHESIS CLASS EXPANSION

The multiple objective problem discussed in the previous chapter improves on the simple problem by allowing us to model assumptions about nature through a hypothesis class H and by letting the coverage constraint depend on the target hypothesis. A drawback to this problem however is that it assumes H is correct ($h^* \in H$) and that the observed responses have no noise ($r_i \in q_i(h^*)$). This is almost never the case in real world problems. For example, recall that in the viral marketing application H models our assumption about the target group and how groups respond to advertising actions. Assuming $h^* \in H$ corresponds to an assumption that the target group exactly matches one of a known set of groups. Similarly, assuming $r_i \in q_i(h^*)$ corresponds to an assumption that the feedback we receive from advertising actions exactly matches our assumptions about how groups respond.

In this chapter we consider a further generalization of interactive submodular set cover which relaxes this assumption. This problem no longer assumes $h^* \in H$ and instead allows responses to be arbitrary. With arbitrary responses it no longer makes sense to require $F_{h^*}(S) = F_{h^*}(Q \times R)$ since we don't have access to F_{h^*}. Instead we require $F_h(S) = F_h(Q \times R)$ for all $h \in H$ which are "close" to h^* (as measured through the observed question-response pairs). This generalization allows us to handle interactive submodular set cover problems with noise. As a special case we derive results for noisy query learning.

Another difficulty with the multiple objective problem is the greedy algorithm computes sums over the hypothesis class. This method is therefore only practical for relatively small hypothesis classes. The generalization we introduce in this chapter can be interpreted as incorporating noise by *implicitly* expanding the hypothesis class. With this technique it is sometimes possible to greatly reduce the size of the explicit representation of the hypothesis class and in some cases eliminate it entirely. This avoids the practical difficulty of storing and summing over the hypothesis class. This idea is discussed further in Section 4.4.

4.1 Problem

We propose defining closeness between h and h^* in terms of additional submodular, monotone functions $G_h(S)$ defined over question-response pairs. In particular, we require that the covering constraint $F_h(S) = F_h(Q \times R)$ is satisfied for all h such that $G_h(S^*) < \kappa$ where κ is a known constant and S^* is the unknown set of all question-response pairs induced by h^*, $S^* \triangleq \bigcup_{q \in Q, r \in q(h^*)} \{(q, r)\}$. S^* is unknown so we cannot directly compute $G_h(S^*)$; however, the question-response pairs we observe are a subset of S^* with which we can reason about $G_h(S^*)$. Intuitively, G_h can be thought of as computing the distance from h to h^* as measured through the question-response pairs allowed by h^*. κ determines which hypotheses are close enough.

We make the assumption that for any h, (G_h, κ) is adversarially uncoverable. We note that a sufficient condition to ensure (G_h, κ) is adversarially uncoverable is to ensure that for any S, q there is some r with $G_h(S + (q, r)) = G_h(S)$. In other words, it sufficient for there to be a response to every question for which G_h does not increase. This is therefore a natural restriction for a function measuring distance. For example, we can define $G_h(S)$ to count the "mistakes" made with respect to h.

$$G_h(S) = |\{(q, r) \in S : r \notin q(h)\}|$$

With this objctive $G_h(S + (q, r)) = G_h(S)$ so long as $r \in q(h)$. Therefore assuming $q(h) \neq \emptyset$ for every $q \in Q$, (G_h, κ) is adversarilly uncoverable for any positive κ.

Interactive Submodular Set Cover (Noisy Version)

Given:

- Hypothesis class H (doesn't contain h^*)
- Query set Q, response set R with known non-empty $q(h) \subseteq R$ for $q \in Q$, $h \in H$ and unknown non-empty $q(h^*) \subseteq R$ for $q \in Q$
- Submodular, monotone $F_h : 2^{Q \times R} \to \mathbb{R}_{\geq 0}$ for every $h \in H$
- Positive threshold κ and submodular, monotone $G_h : 2^{Q \times R} \to \mathbb{R}_{\geq 0}$ for every $h \in H$ with (G_h, κ) adversarially uncoverable

- Modular, positive query cost function c defined over Q

Protocol: At step i algorithm asks $q_i \in Q$, receives a response $r_i \in q_i(h^*)$ from nature

Goal: Achieve $F_h(\bigcup_{j \leq i}\{(q_j, r_j)\}) = F_h(Q \times R)$ for all h such that $G_h(S^*) < \kappa$ using minimal cost $\sum_{j \leq i} c(q_i)$. Here $S^* \triangleq \bigcup_{q \in Q, r \in q(h^*)}\{(q, r)\}$

By setting $\kappa = 1$ and using $G_h(S) \triangleq I(h \notin V(S))$ we recover a variation of the multiple objective interactive submodular set cover problem. In this case satisfying the covering constraint for all h that agree with h^* corresponds exactly to satisfying the covering constraint for all h such that $G_h(S^*) < \kappa$. This is equivalent to the original noise free problem if we include the assumption $G_h(S^*) < \kappa$ for at least one h (i.e. h^* agrees with some $h \in H$). We will show the algorithm we propose is approximately optimal regardless of whether this assumption is made.

We can interpret this more general problem as using an extended notion of version space such that a hypothesis h is no longer immediately eliminated as soon as a question-response pair (q_i, r_i) with $r_i \notin q_i(h)$ is observed. Instead, a hypothesis h is only eliminated from consideration when $G_h(S) \geq \kappa$. Different κ and G_h correspond to different notions of version space.

4.2 Greedy Algorithm and Analysis

Our approach to the noisy problem follows the same general approach as for the noise free multiple objective problem. We define a composite objective function $\bar{F}(S)$ such that (1) covering $\bar{F}(S)$ is necessary and sufficient for the original problem and (2) $\bar{F}(S)$ is still submodular, monotone. We can then reduce the noisy, multiple objective problem to the simple problem. For any G_h, define $G_{h,\kappa}(S) \triangleq \min(G_h(S), \kappa)$. Define the following objective

$$\bar{F}(S) \triangleq \sum_{h \in H}((\kappa - G_{h,\kappa}(S))F_h(S) + G_{h,\kappa}(S)F_h(Q \times R))$$

We show that for worst case h^* ensuring $\bar{F}(S) = \bar{F}(Q \times R)$ is necessary and sufficient for ensuring $F_h(S) = F(Q \times R)$ for all h with $G_h(S^*) < \kappa$.

Lemma 4.1. *For an algorithm to ensure for any h^* that $F_h(S) \geq F_h(Q \times R)$ for all h such that $G_h(S^*) < \kappa$ it is both necessary and sufficient to ensure $\bar{F}(S) = \bar{F}(Q \times R)$ The condition remains sufficient and necessary if the algorithm assumes $G_h(S^*) < \kappa$ for some h.*

Proof. $\bar{F}(S) = \bar{F}(Q \times R) = \sum_h \kappa F_h(Q \times R)$ iff $F_h(S) = F(Q \times R)$ for all h such that $G_h(S) < \kappa$. To see this condition is sufficient note that $G_h(S) \leq G_h(S^*)$. To see this condition is necessary note that for any h, S, with $G_h(S) < \kappa$, there is some h^* with $G_h(S^*) < \kappa$. This follows from our assumption that (G_h, κ) is adversarially uncoverable: there must be an oracle T which can answer any set of questions $\hat{Q}$ so that $G_h(S+T(\hat{Q})) < \kappa$. We can then construct h^* with $q(h^*) = \{T(q)\}$ for all questions not answered in S and this h^* ensures $G_h(S^*) < \kappa$. We cannot therefore eliminate any h with $G_h(S) < \kappa$. The constructed h^* does not violate the assumption that $G_h(S^*) < \kappa$ for some h so the condition remains necessary under this assumption. □

We now show $\bar{F}(S)$ is submodular. We first show a more general purpose lemma for combining two submodular functions F and G through interpolation.

Lemma 4.2. *Let $F(S)$ be a monotone non-decreasing submodular function ranging between 0 and α. Let $G(S)$ be a monotone non-decreasing submodular function ranging between 0 and κ. Then*

$$\bar{F}(S) \triangleq (\kappa - G(S))F(S) + G(S)\alpha$$

is a monotone non-decreasing submodular function.

Proof. As a short hand we use $\delta_S(F, x) \triangleq F(S + x) - F(S)$. We show $\bar{F}(S)$ is monotone non-decreasing by showing that for any A and x, $\delta_A(\bar{F}, x) \geq 0$.

$$\begin{aligned}\delta_A(\bar{F}, x) &= \kappa\delta_A(F, x) + \delta_A(G, x)\alpha + G(A)F(A) - G(A + x)F(A + x) \\ &= (\kappa - G(A))\delta_A(F, x) + \delta_A(G, x)(\alpha - F(A + x))\end{aligned}$$

We see that each of these terms is positive so long as F and G are monotone non-decreasing and range between 0 and α and 0 and κ respectively. We now show $\bar{F}(S)$ is submodular by

showing, for any $A \subseteq B$ and x, $\delta(\bar{F}, A, x) \geq \delta(\bar{F}, B, x)$. As before we have

$$\delta_B(\bar{F}, x) = (\kappa - G(B))\delta_B(F, x) + \delta_B(G, x)(\alpha - F(B + x))$$

Each term in this expression is less than the corresponding term in $\delta_A(\bar{F}, x)$. □

We note Lemma 4.2 does not follow trivially from any of the standard results for combining submodular functions of which we are aware. In particular, the term $(\kappa - G(S))F(S)$ is not by itself submodular so the result doesn't follow from the submodularity of the sum of two submodular functions.

Corollary 4.1. *$\bar{F}(S)$ is submodular monotone non-decreasing whenever F_h and G_h are submodular monotone non-decreasing for all $h \in H$.*

Proof. The result follows from Lemma 4.2 and Propositions 1.6 and 1.7. □

We are now prepared to show optimality of the greedy algorithm applied to $\bar{F}(S)$. Note that we show the result both for an unrestricted adversary and for the case where we assume $G_h(S^*) < \kappa$ for some h.

Theorem 4.1. *Assume $F_h(S)$, $G_h(S)$ are integer valued for every h and κ is an integer. If there is an algorithm for the noisy interactive submodular set cover problem $(Q, R, H, \{F_h\}, \kappa, \{G_h\}, c)$ with worst case cost C^* then applying the worst case greedy algorithm to $\bar{F}(S)$ incurs cost C*

$$C \leq C^*(1 + \ln(\kappa \sum_h F_h(Q \times R)))$$

The result also holds if we assume $G_h(S^*) < \kappa$ for some h.

Proof. By 4.1 it is necessary for an algorithm for $(Q, R, H, \{F_h\}, \kappa, \{G_h\}, c)$ to cover $\bar{F}(S)$. Therefore an algorithm with worst case cost C^* on $(Q, R, H, \{F_h\}, \kappa, \{G_h\}, c)$ implies an algorithm for the interactive submodular set cover problem $(Q, R, \bar{F}, c)$ with worst case cost C^* . The result for the unrestricted adversary case then follows from Theorem 2.1.

Define $\bar{G} \triangleq \sum_h G_h(S)$. If nature always answers such that $G_h(S^*) < \kappa$ for some $h \in H$ then nature is $(\bar{G}, \kappa|H|)$-restricted. The converse is also true. If for every h, (G_h, κ) is

adversarially uncoverable then $(\bar{G}, \kappa|H|)$ is also adversarially uncoverable. Also note that $\bar{G}(S) \geq \kappa|H|$ implies $\bar{F}(S) = F(Q \times R)$. Therefore an algorithm which has worst case cost C^* on $(Q, R, H, \{F_h\}, \kappa, \{G_h\}, c)$ under the assumption $G_h(S^*) < \kappa$ for some $h \in H$ is also an algorithm for $(Q, R, \bar{F}, c)$ which, under the assumptions in Theorem 2.3, also has worst case cost C^* . The result for the restricted adversary case then follows from Theorem 2.3. □

4.3 Connection to Noisy Active Learning

Recall that the simple version of interactive submodular set cover can be used for exact active learning and the multiple objective version can be used for approximate active learning. In this section we show the noisy version of interactive set cover can also be used for active learning in the presence of adversarial noise. Before we define the problem we define the notion of a submodular loss.

Definition 4.1. *Define a* submodular loss *to be a function $\ell(h, S)$ for $h \in H$, $S \subseteq (Q \times R)$ such that*

- *$\ell(h, S)$ is a submodular function of S for any $h \in H$*
- *$(\ell(h, S), \kappa)$ is adversarially uncoverable for any h and any positive κ*

We now define a noisy active learning problem where the goal is to find a hypothesis h with $\ell(h, S^*) < \kappa$.

Noisy Active Learning

Given:

- Hypothesis class H (doesn't contain h^*)
- Query set Q, response set R with known non-empty $q(h) \subseteq R$ for $q \in Q$, $h \in H$ and unknown non-empty $q(h^*) \subseteq R$ for $q \in Q$
- Submodular loss $\ell(h, S)$
- Distance measure $d(h, h')$ over H

- Positive threshold κ
- Modular, positive query cost function c defined over Q

Protocol: At step i algorithm asks $q_i \in Q$, receives a response $r_i \in q_i(h^*)$ from nature

Goal: Using a minimal cost sequence of questions identify an $h \in H$ with $\ell(h, S^*) < \kappa$

Here $S^* \triangleq \bigcup_{q \in Q, r \in q(h^*)} \{(q, r)\}$.

Note that we assume we are given a distance function d but this distance function is not used in the goal. We will shortly make additional assumptions about the relationship between d and ℓ which make the distance function a useful piece of knowledge for the algorithm. We leave these assumptions out of the formal problem statement since the assumptions we use vary between our different approximation results.

As a concrete example, consider again learning a binary classifier. Let Q be a set of unlabeled data. Define $q(h) = \{0\}$ or $q(h) = \{1\}$ to be the label assigned to q by a classifier $h \in H$. We can then define d to be

$$d(h, h') = \sum_{q \in Q} I(q(h) \neq q(h')) \tag{4.1}$$

and ℓ to be

$$\ell(h, S) = \sum_{(q,r) \in S} I(r \notin q(h)) \tag{4.2}$$

These functions have the standard interpretations: $d(h, h')$ is the number of data points on which h and h' disagree and $\ell(h, S)$ is the number of mistakes h makes on S.

We note a small technical difference between this setting and standard active learning settings is that nature is allowed to respond arbitrarily, so h^* is not restricted to always label the same point the same way. In other words, $q(h^*)$ may be of size greater than 1 so if we ask q twice we may receive different response. This is the case even if $|q(h)| = 1$ for every $h \in H$. However, this difference does not seem to have an effect on the worst-case analysis: if asked the same question multiple times an adversary will always respond identically so as to not provide any additional information.

We define two different notions of compatibility between $d(h, h')$ and $\ell(h, S)$ which we will use in our analyses.

Definition 4.2. *Define a submodular loss $\ell(h, S)$ and a distance function $d(h, h')$ to be* distance bound compatible *if for $h \in H$, $h' \in H$, h^* with $q(h^*) \neq \emptyset$*

$$d(h, h') \leq \ell(h', S^*) + \ell(h, S^*)$$

where $S^* \triangleq \bigcup_{q \in Q, r \in q(h^*)} \{(q, r)\}$.

Definition 4.3. *Define a submodular loss $\ell(h, S)$ and a distance function $d(h, h')$ to be* loss bound compatible *if for $h \in H$, $h' \in H$, h^* with $q(h^*) \neq \emptyset$*

$$\ell(h, S^*) \leq d(h, h') + \ell(h', S^*)$$

where $S^* \triangleq \bigcup_{q \in Q, r \in q(h^*)} \{(q, r)\}$.

Note that these are both variations of the triangle inequality relating ℓ and d.

Lemma 4.3. *Assume $|q(h)| = 1$ for every $q \in Q$, $h \in H$. (4.1) and (4.2) are distance bound compatible and loss bound compatible.*

Proof. To show they are distance bound compatible we show that for any q

$$I(q(h) \neq q(h')) \leq |q(h^*) \setminus q(h)| + |q(h^*) \setminus q(h')|$$

The result then follows from summing over every q. If $q(h) = q(h')$ then the l.h.s is 0 and the bound follows trivially. If $q(h) \neq q(h')$ then the l.h.s. is 1. Note in this case $|q(h)| = 1$, $|q(h')| = 1$, and $(q(h) \cap q(h')) = \emptyset$. Because $q(h^*) \neq \emptyset$ there is at least one $r \in q(h^*)$ and we must have $r \notin q(h)$ or $r \notin q(h')$ (or both). The r.h.s. is then at least 1.

To show they are loss bound compatible we show that for any q

$$|q(h^*) \setminus q(h)| \leq I(q(h) \neq q(h')) + |q(h^*) \setminus q(h')|$$

If $q(h) = q(h')$ then the bound holds with equality. If $q(h) \neq q(h')$ then the bound holds because $|q(h^*) \setminus q(h)| - |q(h^*) \setminus q(h')| \leq 1$. □

Define $V_\kappa(S)$ to be the subset of the hypothesis class making fewer than κ errors on S

$$V_\kappa(S) = \{h \in H : \ell(h, S) < \kappa\}$$

This can be thought of as an extended notion of version space. If we assume that there is only one hypothesis h with h with $\ell(h, S^*) < \kappa$ then identifying this h is equivalent to ensuring $|V_\kappa(S)| = 1$. Unfortunately $F(S) = |H \setminus V_\kappa(S)|$ is not a submodular function. To see this note that a hypothesis is not eliminated until we have observed κ errors, so for $\kappa > 1$ the gain of $F(S)$ for a particular question response pair may increase as we ask other questions. Because of this the problem requires a more subtle analysis than the exact learning case. We would also like to handle the case where possibly more than one hypothesis has low error.

We now define a noisy interactive submodular set cover problem to which we will reduce noisy active learning. Define objective functions

$$F_{h,\epsilon}(S) \triangleq \sum_{h':d(h,h')\geq\epsilon} \min(\ell(h', S), \kappa) \tag{4.3}$$

and

$$G_h(S) \triangleq \ell(h, S) \tag{4.4}$$

Lemma 4.4. *(4.3) is submodular, monotone, non-decreasing for any ϵ*

Proof. From Proposition 1.7 it suffices to show that for any h, $F(S) = \min(\ell(h, S), \kappa)$ is submodular, monotone. $\ell(h, \hat{S})$ is submodular, so this follows from Propsition 1.6. □

The following lemma establishes the equivalence between solving the noisy interactive submodular set cover problem $(Q, R, H, \{F_{h,\epsilon}\}, \kappa, \{G_h\}, c)$ and identifying a ball of diameter less than ϵ containing all hypotheses with error less than κ. This is similar to the Lemma 3.4 for the approximate active learning problem.

Lemma 4.5. *To identify a set $\hat{H} \subseteq H$ such that (1) $\max_{h\in\hat{H},h'\in\hat{H}} d(h, h') < \epsilon$ and (2) $h \in \hat{H}$ for every h with $\ell(h, S^*) < \kappa$ it is necessary and sufficient to solve the noisy interactive submodular set cover problem $(Q, R, H, \{F_{h,\epsilon}\}, \kappa, \{G_h\}, c)$ with (4.3) and (4.4)*

Proof. From Lemma 4.1 an algorithm for $(Q, R, H, \{F_{h,\epsilon}\}, \kappa, \{G_h\}, c)$ must ensure $F_{h,\epsilon}(S) = F_{h,\epsilon}(Q \times R)$ for every $h \in V_\kappa(S)$. In this case $\max_{h\in V_\kappa(S),h'\in V_\kappa(S)} d(h, h') < \epsilon$. Note also that $h \in V_\kappa(S)$ for every h with $\ell(h, S^*) < \kappa$. Therefore solving $(Q, R, H, \{F_{h,\epsilon}\}, \kappa, \{G_h\}, c)$ is sufficient.

We now show that solving $(Q, R, H, \{F_{h,\epsilon}\}, \kappa, \{G_h\}, c)$ is necessary. Consider any algorithm which identifies a $\hat{H}$ as described. We must have $V_\kappa(S) \subseteq \hat{H}$ where S is the set of questions asked by the algorithm. Otherwise some choice of h^* makes $h \notin \hat{H}$ for an h with $\ell(h, S^*) < \kappa$. Taking a subset of $\hat{H}$ only decreases the diameter, so $\max_{h \in V(S), h' \in V(S)} d(h, h') < \epsilon$. Therefore $F_{h,\epsilon}(S) = F_{h,\epsilon}(Q \times R)$ for every $h \in V_\kappa(S)$ and the algorithm also solves $(Q, R, H, \{F_{h,\epsilon}\}, \kappa, \{G_h\}, c)$ □

We now use this lemma to construct an approximation algorithm for noisy active learning. We first show a result for the special case where $d(h, h') \geq 2\kappa$ for every h, h' in H. This condition would hold if, for example, H were constructed via a clustering algorithm which ensured sufficient distance between every hypothesis [Dasgupta et al., 2003] (e.g. by clustering users of a collaborative filtering system). Here we also require ℓ and d are distance bound compatible.

Theorem 4.2. *Assume (1) there is an $h \in H$ with $\ell(h, S^*) < \kappa$, (2) ℓ and d are distance bound compatible, and (3) $d(h, h') \geq 2\kappa$ for every h, h' in H. Assume ℓ is integer valued. If there is an algorithm for the noisy active learning problem $(Q, R, H, \ell, d, \kappa, c)$ with worst case cost C^* then applying the worst-case greedy algorithm to $\bar{F}(S)$ with (4.3), (4.4), $\epsilon = 2\kappa$ identifies an h with $\ell(h, S^*) < \kappa$ using cost*

$$C \leq (1 + 2 \ln \kappa |H|) C^*$$

Proof. If $d(h, h') \geq 2\kappa$ for every h, h' in H, then there can only be one h with $\ell(h, S^*) < \kappa$. To see this note that for any $h \in H$ with $\ell(h, S^*) < \kappa$ and any other h' with $d(h, h') \geq 2\kappa$, we have from Definition 4.2

$$\ell(h', S) \geq d(h, h') - \ell(h, S) > \kappa$$

Under these assumptions and with $\epsilon = 2\kappa$ identifying the set $\hat{H}$ described in Lemma 4.5 is then equivalent to identifying the one $h \in H$ with $\ell(h, S^*) < \kappa$: the set $\hat{H}$ is of size one. Identifying the one h with $\ell(h, S^*) < \kappa$ is therefore also equivalent to solving the noisy interactive submodular set cover problem $(Q, R, H, \{F_{h,2\kappa}\}, \kappa, \{G_h\}, c)$. Assuming there is

an $h \in H$ with $\ell(h, S^*) < \kappa$ is equivalent to assuming $G_h(S^*) < \kappa$ for some h. The theorem then follows from Theorem 4.1. □

We note that interestingly this bound does not require d to obey the triangle inequality (Lemma 4.5 also does not use this). The bound only requires distance bound compatibility (Definition 4.2) between ℓ and d.

We now derive a slightly weaker result for more general hypothesis classes. Here we use the close relationship between finding low diameter balls and low radius balls (as in Theorem 3.2). The bound is slightly weaker because it is a bicriteria approximation. The bound also differs from Theorem 4.2 in that it requires d obey the triangle inequality and both distance bound and loss bound compatibility between ℓ and d as opposed to only distance bound compatibility. We first show that solving $(Q, R, H, \{F_{h,\epsilon}\}, \kappa, \{G_h\}, c)$ is *sufficient* for identifying a $\kappa + \epsilon$ error hypothesis.

Lemma 4.6. *Assume there is some $h \in H$ with $\ell(h, S^*) < \kappa$ and that ℓ and d are loss bound compatible. To identify an $h \in H$ with $\ell(h, S^*) < \kappa + \epsilon$ it suffices to solve the noisy interactive submodular set cover problem $(Q, R, H, \{F_{h,\epsilon}\}, \kappa, \{G_h\}, c)$ with (4.3) and (4.4)*

Proof. After solving $(Q, R, H, \{F_{h,\epsilon}\}, \kappa, \{G_h\}, c)$ we are guaranteed there is at least one $h \in V_\kappa(S)$ with $\ell(h, S^*) < \kappa$ and that $\max_{h \in V_\kappa(S), h' \in V_\kappa(S)} d(h, h') < \epsilon$. Therefore from Definition 4.3 for any $h' \in V_\kappa(S)$

$$\ell(h', S^*) \leq d(h', h) + \ell(h, S) < \kappa + \epsilon$$

It therefore suffices to solve $(Q, R, H, \{F_{h,\epsilon}\}, \kappa, \{G_h\}, c)$ and return any $h \in V_\kappa(S)$. □

This proof only shows that it is sufficient to solve the noisy interactive submodular set cover problem, not that it is necessary. It also does not specify how to pick ϵ. This theorem completes the analysis.

Theorem 4.3. *Assume there is some $h \in H$ with $\ell(h, S^*) < \kappa$ and that ℓ and d are loss and distance bound compatible. Assume ℓ is integer valued. If there is an algorithm for the noisy active learning problem $(Q, R, H, \ell, d, \kappa, c)$ with worst case cost C^* then applying the*

worst-case greedy algorithm to $\bar{F}(S)$ *with (4.3) and (4.4) with* $\epsilon = 4\kappa$ *identifies an* h *with* $\ell(h, S^*) < 5\kappa$ *using cost*

$$C \leq (1 + 2 \ln \kappa |H|) C^*$$

Proof. Let h be the hypothesis returned by the active learning algorithm incurring cost C^* From Definition 4.2 we know that for any other h' with $\ell(h', S^*) < \kappa$, $d(h, h') \leq \ell(h, S^*) + \ell(h, S^*) < 2\kappa$. Therefore if we construct a set $\hat{H}$ containing all $h' \in H$ with $d(h, h') < 2\kappa$ then we are guaranteed that $h' \in \hat{H}$ for every h' with $\ell(h', S^*) < \kappa$. From the triangle inequality for d, this set has diameter less than 4κ. From Lemma 4.5 this implies an algorithm for the multiple objective interactive submodular set cover problem $(Q, R, H, \{F_{h,4\kappa}\}, \kappa, \{G_h\}, c)$ with worse case cost C^*. Assuming there is an $h \in H$ with $\ell(h, S^*) < \kappa$ is equivalent to assuming $G_h(S^*) < \kappa$ for some h. The theorem then follows from Theorem 4.1 and Lemma 4.6. □

This is a bicriteria approximation (similar to Theorem 3.2) because our algorithm only identifies an h with $\ell(h, S^*) < 5\kappa$, but is compared to the optimal algorithm identifying an h with $\ell(h, S^*) < \kappa$.

Dasgupta et al. [2003] study a similar query learning setting with adversarial noise. Under the hypothesis class assumptions of Theorem 4.2, their algorithm gives an $O(\ln |H|)$ approximation. We suspect our added dependence on κ is due to the increased flexibility of ℓ in the more general problem (G_h does not need to be defined in terms of an additive or metric loss function and can be hypothesis dependant). Assuming only that H contains one or more h with $\ell(h, S^*) < \kappa$ (like Theorem 4.3), Dasgupta et al. give an $O(\ln |H|)$ multiplicative approximation for finding one such h, but in this case the algorithm also has an additional additive approximation term which depends on κ and the number of hypotheses near any h. The analysis uses two phases, with the first phase similar to the problem solved in Lemma 4.5. Our Theorem 4.3 is worse in the sense it is a bicriteria approximation, but it is stronger in that it doesn't have an additive approximation term and only depends on $|H|$, not on any structure of H. Recently Golovin et al. [2010b]

gave approximation results for an average-case noisy query learning setting using adaptive submodularity. Bellala et al. [2010] consider related average-case problems.

4.4 Interpretation as Implicit Hypothesis Class Expansion

Consider a scenario where we know $h^* \notin H$ but that h^* is close to the hypothesis in the following simple sense: there is at least one hypothesis $h \in H$ such that $\sum_{(q,r)\in S^*} I(r \notin q(h))$ is small. In other words, there is a hypothesis that mostly agrees with h^*. In this case a simpler approach than the one we have presented would be to expand the hypothesis class in such a way that it includes h^*. In other words, we can explicitly incorporate our assumptions about the noise into the hypothesis class. This is the approach used by Golovin et al. [2010b] in an average-case query learning setting. A drawback to this approach is that this kind of expansion can very quickly grow the hypothesis class size. For example, if we are learning a binary classifier and we wish to expand H to include all hypotheses that are at most κ away from an $h \in H$, then in the worst case we end up increasing the size of H by a factor of $\binom{|Q|}{\kappa}$. If our original approximation ratio is $O(\ln |H|)$ then after expansion we get a ratio of at least $O(\kappa \ln(|Q||H|/\kappa))$.

The noisy interactive submodular set cover problem can be interpreted as performing a similar sort of expansion *implicitly*. Instead of explicitly representing the expanded hypothesis class, we introduce functions $G_h(S)$ which identify when h^* has fallen out of the implicit expansion around h. In this way we are able to derive bounds with approximation ratios of the form $O(\ln(\kappa|H|))$ (with the κ *inside* of the logarithm). Not only can implicitly representing the hypothesis class in this way improve the approximation ratio, it can also make implementing the worst-case greedy algorithm much more practical. To implement the worst-case greedy algorithm we only need to sum over the explicit portion of the hypothesis class and evaluate G_h.

An interesting special case of the noisy problem is when $|H| = 1$ so that the hypothesis class is entirely implicit: our assumptions about nature are completely captured by a single submodular function G and a threshold κ. Below we describe this problem formally.

Interactive Submodular Set Cover (Implicit Hypothesis Class Version)

Given:

- Query set Q, response set R
- Submodular, monotone objective $F : 2^{Q\times R} \to \mathbb{R}_{\geq 0}$
- Closeness threshold κ and submodular, monotone objective $G : 2^{Q\times R} \to \mathbb{R}_{\geq 0}$ with (G, κ) adversarially uncoverable
- Modular, positive query cost function c defined over Q

Protocol: At step i algorithm asks $q_i \in Q$, receives a response r_i from nature

Assume: Nature always responds such such that $G(S) < \kappa$

Goal: Achieve $F\bigcup_{j\leq i}\{(q_j, r_j)\}) = F(Q \times R)$ using minimal cost $\sum_{j\leq i} c(q_i)$

Clearly this is a special case of the multiple objective noisy problem. However, we find this problem interesting since it illustrates how in some cases we can completely avoid explicit hypothesis class representation. In many real world problems it is unrealistic to assume the hypothesis class is small enough to represent explicitly. In these problems, it may be possible to design a (G, κ) which represents the hypothesis class implicitly. If this is the case, we need only to be able to efficiently evaluate G in order to implement the worst-case greedy algorithm for $\bar{F}$.

4.5 Interpretation as Monotone Circuits of Constraints

One interpretation of noisy interactive set cover is in terms of a boolean circuit of submodular cover constraints. For every h we want either $F_h(S) = F_h(Q \times R)$ or $G_h(S) \geq \kappa$. Figure 4.1 shows the boolean circuit encoding this. This circuit is monotone (i.e. it only involves AND and OR gates). A key part of our analysis is showing that this can be reduced to a single constraint $\bar{F}(S) = \bar{F}(Q\times R)$. An interesting corollary of our analysis and in particular Lemma 4.2 is that in fact *any* monotone boolean circuit of constraints to a single constraint.

Lemma 4.7. *Given $i = 1 \dots n$ constraints $F_i(S) \geq \alpha_i$ for normalized, monotone submodular F_i and any monotone boolean circuit over these constraints, there is a normalized, monotone submodular $\bar{F}(S)$ and $\bar{\alpha}$ such that $\bar{F}(S) \geq \bar{\alpha}$ iff the circuit evaluates to true.*

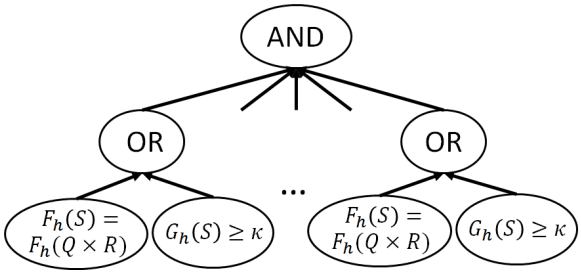

Figure 4.1: Noisy interactive submodular set cover as a circuit.

Proof. We first convert the constraints $F_i(S) \geq \alpha_i$ into constraints $\hat{F}_i(S) = \alpha_i$ where $\hat{F}_i(S) = \min(F_i(S), \alpha_i)$. $\min(F_i(S), \alpha_i)$ is submodular and ranges between 0 and α_i.

We now show how to reduce the OR of two constraints $(\hat{F}_i(S) = \alpha_i) \vee (\hat{F}_j(S) = \alpha_j)$ to a single constraint $\bar{F}(S) = \bar{\alpha}$ where $\bar{F}$ ranges between 0 and $\bar{\alpha}$. Define

$$\bar{F}(S) = (\alpha_i - \hat{F}_i(S))\hat{F}_j(S) + \hat{F}_i(S)\alpha_j$$

It is not hard to see that indeed $\bar{F}(S) = \alpha_i\alpha_j$ iff $(\hat{F}_i(S) = \alpha_i) \vee (\hat{F}_j(S) = \alpha_j)$. Lemma 4.2 shows that $\bar{F}$ is submodular and monotone.

We now show how to reduce the AND of two constraints $(\hat{F}_i(S) = \alpha_i) \wedge (\hat{F}_j(S) = \alpha_j)$. Define

$$\bar{F}(S) = \hat{F}_i(S) + \hat{F}_j(S)$$

as in Krause et al. [2008a]. $\bar{F}(S) = \alpha_i + \alpha_j$ iff $(\hat{F}_i(S) = \alpha_i) \wedge (\hat{F}_j(S) = \alpha_j)$. $\bar{F}$ is submodular and monotone since it is the sum of monotone, submodular functions.

In both cases $\bar{F}(S) \geq \bar{\alpha} \iff \bar{F}(S) = \bar{\alpha}$ since $\bar{F}$ ranges from 0 to $\bar{\alpha}$. Furthermore, AND and OR combinations are all we need, since any monotone circuit can be written in terms of these operators. □

This lemma can be seen as generalizing the result of Krause et al. [2008a] for converting the AND of a set of constraints to a single constraint. With this lemma we can solve (interactive) submodular set cover problems with constraints given by arbitrary monotone

circuits. The $\bar{F}$ and $\bar{\alpha}$ derived are integer valued when all F_i and α_i are integer valued. We also need for $\bar{\alpha}$ to be small for the reduction to be useful. Assume for simplicity that $\alpha_i = \alpha$ for all i and the circuit is written in disjunctive normal form (this is always possible). In this case, $\bar{\alpha} \leq c_1 \alpha^{c_2}$ where c_1 is the number of clauses in the circuit and c_2 the size of the largest clause in the circuit (the OR nodes multiply and the AND nodes add). The approximation ratio given by the greedy algorithm is then $\ln \bar{\alpha} \leq c_2 \ln c_1 \alpha$. We therefore expect this reduction to be useful if c_2 is small (in our noisy problem it is 2). This general reduction may be of interest outside of our interactive optimization setting. For example, in an advertising application we can require that each of several demographics is influenced by at least one of several advertising campaigns.

4.6 Application to Movie Recommendation

As an example real world application of our theoretical results, we present a website we have developed for movie recommendation which minimizes the total cost of learning and recommending. The standard approach to movie recommendation uses collaborative filtering Breese et al. [1998]. We take a different, complimentary approach. Instead of learning from a user's fixed rating history, we directly asking the user questions. Furthermore, instead of trying to learn a comprehensive model of the user's taste, we try to learn what the user wants to watch *right now*. We think this approach is well suited for streaming services.

4.6.1 System Design

We use the Netflix API to retrieve a catalog of all movies and TV shows available through Netflix's Watch Instantly service (approximately 11000 titles). We set our hypothesis class H to be this set of available titles. With this H, assuming $h^* \in H$ corresponds to assuming that the user wants to watch a single title which is in the set of available titles. Through appropriate choice of G_h we can relax this assumption: the user may want to watch a movie that is not available in our catalog but which is at least similar to an available title.

In our query set Q we use four types of questions: (1) genre questions such as "Do you want to watch something from the Comedy genre?", (2) release year questions such as "Do you want to watch something from the 90s?", (3) runtime questions such as "Do you

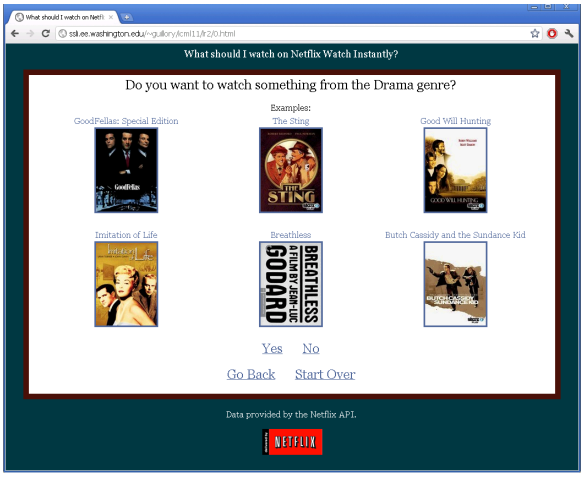

Figure 4.2: Screenshot of the movie recommendation website.

want to watch something under an hour long?", and (4) cast and director questions such as "Do you want to watch a movie featuring Tom Hanks?". In addition, we include in Q a recommendation action for each title. We assume these actions have no feedback and assign them cost .1 that of a question.

We experimented with three different versions of our website which use different combinations of objective functions F_h and G_h. Each version of the website is created by solving the corresponding noisy interactive set cover problem using the worst-case greedy algorithm. In our current implementation, we solve the problem for all possible sequences of responses and use the resulting decision tree to generate a static web page. With implementation tricks [Minoux, 1978] for speeding up the greedy algorithm, we've found that writing the web page to disk usually dominates the run time. Each question is presented to the user on a separate page along with a set of 6 positive examples (displayed as box art thumbnail images) taken from the current version space. Figure 4.2 shows the the question asking interface. Recommendations are presented in lists with box art and a plot synopsis.

Our first, simplest version uses $\kappa = 1$ and $G_h(S) \triangleq I(h \notin V(S))$. For the covering objective, this first version uses $F_h(S) = 1$ iff S includes the recommendation action corre-

sponding to h. This very simple problem does not use the full power of noisy interactive set cover and, in fact, is equivalent to query learning with membership and equivalence queries.

Our second version uses $\kappa = 2$ and $G_h(S) \triangleq \sum_{(q,r)\in S} I(r \notin q(h))$. As compared to the first version, the second version does not eliminate a title from consideration until the user's responses have disagreed with this title twice.

Our third version again uses $\kappa = 2$ but uses a more complicated modular function for G_h which encodes domain knowledge about the problem. For example, if the user responds "Yes" to a genre question then our $G_h(S)$ increases by 2 for h that are not in that genre or any related genres but only by 1 for h that are not in the genre but are in related genres. We use similar heuristics for release year and runtime questions.

The third version also uses a more complicated covering objective F_h. We use $F_h(S) = 1$ if S contains either the recommendation action corresponding to h or a recommendation action corresponding to a title h' which *covers* h. We say a title h' covers h if: (1) h is in the list of titles similar to h' available from the Netflix API, (2) h' is tagged with all genres h is tagged with, and (3) the average user rating for h' is greater than or equal to that for h. This essentially defines a directed graph over titles. We also use this same directed graph to create a small list of related titles displayed with each recommendation. In this way, titles which are relevant but covered by another recommendation are often still shown to the user but with less emphasis. For consistency we display these same related titles in all three versions of the website.

4.6.2 User Study Results

We conducted a small user study comparing the three different versions of our website. We asked each user to try each version of the website (presented in a different random order for each user) and then fill out a short survey. For each website, the survey asked users how strongly they agree with four statements. 1. The website was useful for finding things to watch. 2. The website's questions were easy to answer. 3. The website's recommendations matched my responses to the questions that were asked. 4. The website asked the right number of questions before recommending things to watch. We also asked the users to

Table 4.1: Average survey responses.

Statement	Website 1	Website 2	Website 3
1	3.86 (0.90)	3.52 (1.08)	3.05 (1.02)
2	4.14 (0.80)	4.00 (0.81)	4.05 (0.78)
3	3.95 (0.97)	3.58 (1.00)	3.21 (1.07)
4	3.44 (1.15)	2.83 (1.13)	2.90 (1.16)

describe the differences between the sites and give suggestions.

We received 59 survey responses. Table 4.1 shows the average responses. 5 is Strongly Agree, 1 is Strongly Disagree and standard deviations are shown in parentheses. The variance was relatively high, but there are some large significant differences in particular Website 1 vs Website 3 on Statements 1 and 3. Surprisingly, there is an average preference for the simplest site, Website 1, which assumes $h^* \in H$. 45/59 participants rated Website 1 as being more useful or as useful compared to the other websites. 35/59 rated Website 2 as being more or as useful and only 23/59 rated Website 3 as being more or as useful.

Several users felt that Website 2 asked too many questions. One user commented that Website 2 asked questions that were too similar to previously asked questions. These were issues we hoped to address with the more complicated G_h and F_h objectives used in Website 3, and this partially worked; users did not think that Website 3 asked too many questions.

In fact, because its F_h objective assigns larger gains to certain titles, Website 3 often recommended a few titles early and followed up with further questions and recommendations. A few users specifically mentioned that they liked this. More users however commented they received poor or unexpected recommendations too early and gave Website 3 lower scores because of this. Some users did not seem to realize that by clicking "More" they would be asked further questions and given more recommendations. To remedy this, about midway through the study we changed the UI to make this link more apparent, but this change did not seem to improve perception of Website 3, and it was still not clear how many users chose to continue after the first recommendation.

Since users tended to prefer the simpler website, one could interpret these results as evidence against the practical utility of our more flexible noisy version of the problem. However, in light of the user comments we received explaining their issues with the more complex websites, we think these results are more indicative of problems designing objective and cost functions by hand. One very promising direction for future work would be to learn the objectives. The user feedback also suggests some specific ways our theory may be useful in future systems. For example, our results suggest that, if any noise is allowed, a more complex model than additive zero-one error is necessary as this simple model resulted in the system asking too many questions. Several users also reported they wanted the ability to respond "Maybe". Incorporating such a response would be problematic in a noise free query learning setting: if we include "Maybe" in the set of allowed responses $q(h)$ for every q and h then then the worst-case gain of every learning action is 0. However, with our approach we can treat these responses as noise.

Chapter 5

BATCH ACTIVE LEARNING

Recall that the simple approach to active learning through submodular optimization (Section 1.4) is to set the ground set to be some collection of unlabeled data, define some submodular objective that measures informativeness, then use an algorithm for submodular set cover or submodular function maximization to select a labeled set. A drawback of this approach is that it is necessarily batch: it selects all of the labeled points at a time. The previous three chapters introduced interactive submodular optimization problems which overcome this drawback. However, for some applications it actually makes sense to select labels in batch. For example, if acquiring a label involves running a time consuming experiment, it may be possible to acquire multiple labels in parallel. In this case it may be significantly cheaper to select labels in a batch.

A different drawback to the simple approach is that it is not clear how to choose a good submodular informativeness measure. There are many reasonable such objectives. Moreover, if we choose some arbitrary heuristic informativeness measure then it is not clear how the optimality guarantees of submodular optimization relate to the learning problem: ideally our informativeness measure should be such that when it is maximized we are guaranteed a low error classifier. In the previous chapters we have shown that in the interactive setting it is possible to give guarantees of this sort. We gave submodular objective functions such that solving a interactive submodular optimization problem corresponds in a precise way to a learning problem (Sections 2.7, 3.4, and 4.3). In this section we pursue similar guarantees for the batch setting.

Aside from its practical value, the batch setting is interesting from a theoretical perspective because it is very different from traditional active learning (such as the settings considered in the previous chapters). Many traditional active learning algorithms are based on efficiently searching the hypothesis space by using a series of label queries to divide up the

space. Batch active learning algorithms do not have the option to query labels sequentially, so they must instead take advantage of more data set specific properties such as cluster structure.

It is possible in some cases to directly apply the same objectives we proposed for the interactive setting to batch learning. Section 3.6.3 discussed how when $F_h(S)$ is fully determined by the set of questions asked (as opposed to both the questions and responses), we can derive an approximately optimal batch algorithm for interactive submodular set cover via a reduction to the standard (non-interactive) submodular set cover problem. If we assume $|q(h)| = 1$ (queries map hypotheses to *single* responses) than this strategy can be used in conjunction with the exact and approximate learning objectives (Sections 2.7 and 3.4). However, this strategy fails for the noisy learning problem (Section 4.3). Here $F_h(S)$ is not fully determined by the set of questions asked even when $|q(h)| = 1$. It is also not clear how to use this strategy with implicit hypothesis class representations or expansions like those discussed in Chapter 4.

In this chapter we give error bounds for batch active learning which *do not* make realizability assumptions or require explicit hypothesis class representation. To give such error guarantees we will need to make assumptions about the data representation and the learning method used to make predictions. We propose a setting where the data is represented through symmetric submodular function and predictions are made through constrained minimization of this function. In this setting we derive error bounds which relate prediction error to the labeled set and regularization function. We then derive an active learning algorithm which approximately minimizes this bound. To make this abstract setting more clear we begin by describing a special case: the minimum-cut prediction method for learning on graphs [Blum and Chawla, 2001].

5.1 Semi-Supervised Learning with Graph Cuts

In graph-based semi-supervised learning we are given as input a partially labeled data set represented as an undirected graph. We assume the data set of n data points V with the graph specified by a non-negative, symmetric edge weight matrix W. We assume that the nodes of the graph are labeled according to an unknown binary label vector $y \in \{0,1\}^V$. We

do not know y but we do know the labels y_L for a labeled subset $L \subseteq V$. Our goal is then to predict the unseen values $y_{V \setminus L}$ using the observed values y_L and the graph structure. We use $\hat{y}$ to refer to the learning algorithm's prediction.

Clearly the problem we have described is impossible for general y–if the labels are random then we can predict no better than chance. The assumption made by the minimum cut prediction method which makes learning possible is that *cut value* of the cut induced by y is small. More formally, define

$$\phi(y) = \sum_{i,j \neq i} W_{i,j} |y_i - y_j| \tag{5.1}$$

The assumption that $\phi(y)$ is small is a natural homophily assumption: we assume that, on average, differently labeled points are only weakly connected in the graph. Note that we can rewrite $\phi(y)$ as $\Gamma(V_{y=0}) = \Gamma(V_{y=1})$ where Γ is the symmetric submodular graph cut function (1.3) and $V_{y=0}$ ($V_{y=1}$) is the negatively (positively) labeled subset of V. Therefore this assumption is an example of using a symmetric submodular function as a sort of regularization term. Γ models our prior beliefs about the graph partition induced by y.

Under the assumption that $\phi(y)$ is small a natural prediction method is to predict labels $\hat{y}$ which are consistent with y_L and minimize $\phi(\hat{y})$.

$$\operatorname*{argmin}_{\hat{y}: \hat{y}_L = y_L} \phi(\hat{y})$$

Blum and Chawla [2001] show that this prediction can be computed efficiently via an s-t minimum cut computation in a graph derived from W. Specifically the graph connects all positively labeled points in L to the source s and all negatively labeled points in L to the sink t, both with infinite weight. Blum et al. [2004] show that if we assume the labeled set L is selected uniformly at random from V then this prediction method is guaranteed to achieve an error rate on the order of $O(\phi(y)/|L|)$.

The minimum cut prediction method we have described simple, intuitive, and has a nice theoretical justification. However, the assumption that $\phi(y)$ is small may not be appropriate for every application. Not every data set can be naturally represented as an edge weighted graph, for example. Also, it is not clear how to perform active learning in this setting: the error bound of Blum et al. [2004] is only valid if L is selected uniformly at random.

Cesa-Bianchi et al. [2010b] give a batch active learning algorithm for tree graphs with low cut value which is optimal up to constant factors. Cesa-Bianchi et al. [2010b] also show that their algorithm for active learning on tree graphs minimizes our proposed error bound (Theorem 5.1). When used on general graphs via spanning trees, this method gives good empirical performance [Cesa-Bianchi et al., 2010a] but does not have any (known) theoretical guarantees. Afshani et al. [2007] consider active learning on graphs with sequential (e.g. non-batch) strategies. They give an approximately optimal algorithm for exact learning in this setting and an algorithm for approximate learning when the cut is also balanced. An interesting separate line of work has considered mistake bounds for an online setting in which the labels for the nodes of a graph are predicted sequentially in an adversarially selected order [Herbster and Lever, 2009, Cesa-Bianchi et al., 2009].

5.2 Symmetric Submodular Functions as Regularizers

Instead of assuming the data set is represented as a graph with small cut value, we assume our prior knowledge about the data is in form of a symmetric submodular function $\Gamma(S)$. In particular, we assume that for the true, unknown labels y, $\Gamma(V_{y=0}) = \Gamma(V_{y=1})$ is small. The graph cut smoothness assumption is a specific example of this. For other different kinds of data other symmetric submodular functions make sense. For example, say that the data set was not in the form of a graph but rather a hypergraph (a graph in which edges connect more than two nodes). In this case we could set $\Gamma(S)$ to be the hypergraph cut function, which is still symmetric and submodular. Another example, say that the ground set V is a set of (possibly correlated) random variables. We could then set $\Gamma(S)$ to be symmetric mutual information. In this case the assumption that $\Gamma(V_{y=0})$ is small corresponds to the assumption that negatively labeled nodes provide little information about the positively labeled nodes. We assume that Γ is non-negative and normalized ($\Gamma(\emptyset) = 0$).

The goal of our learning algorithm is to ensure that prediction error $||y - \hat{y}||^2$ is small so long as our assumption that $\Gamma(V_{y=0})$ is small holds. Note that since y and $\hat{y}$ are both binary vectors $||y - \hat{y}||^2$ just counts the number of prediction errors. In the batch active learning setting, learning consists of two rounds. We first select a labeled set L, then receive the observed labels y_L, then finally make a prediction $\hat{y}$. We formalize this setting below.

Batch Active Learning with Symmetric Submodular Functions

Given: Symmetric submodular function Γ defined over V

Protocol:

- Learning algorithm selects labeled set $L \subseteq V$
- Nature reveals labels $y_L \in \{0,1\}^L$
- Learning algorithm predicts labels $\hat{y} \in \{0,1\}^V$

Goal: Ensure $||y - \hat{y}||^2$ is small when $\Gamma(V_{y=0})$ is small

We overload notation from the graph cut setting to make the connection to this setting clear. In this more general setting we define

$$\Phi(y) \triangleq \Gamma(V_{y=1}) = \Gamma(V_{y=0})$$

This Φ is the same as (5.1) when Γ is the cut function and is the number of hypergraph edges cut by y when Γ is the hypergraph cut function.

We propose the following function, $\Psi(L)$ as a measure of the quality of a potential labeled set.

$$\Psi(L) = \min_{S \subseteq (V \setminus L): S \neq \emptyset} \frac{\Gamma(S)}{|S|}$$

This is related to the strength of a graph [Cunningham, 1985]. Intuitively, if $\Psi(L)$ is large then an adversary must incur a large "cut" value (as measure by Γ) in order to separate a large number of nodes from L. Ψ therefore makes sense as an objective we might want to maximize when selecting the labeled set L. Figure 5.1 illustrates a bad and a good labeled set according to this objective.

As it is written $\Psi(L)$ is undefined for $L = V$. In this case there is no non-empty set T with $T \subseteq (V \setminus L)$. For mathematical convenience we define $\Psi(V) \triangleq \max_{S \subset V} \Psi(S)$. We note that our proofs actually work no matter how we define $\Psi(V)$ so long as $\Psi(V) \geq \max_{S \subset V} \Psi(S)$

We first show that it is possible to bound prediction error in terms of $\Psi(L)$.

Theorem 5.1. *For any non-negative, normalized symmetric submodular* $\Gamma(S)$ *and any* $L \subseteq V$ *with* $\Psi(L) > 0$, $\hat{y}$ *with* $\hat{y}_L = y_L$,

$$||y - \hat{y}||^2 \leq \frac{\Gamma(V_{y=0}) + \Gamma(V_{\hat{y}=0})}{\Psi(L)}$$

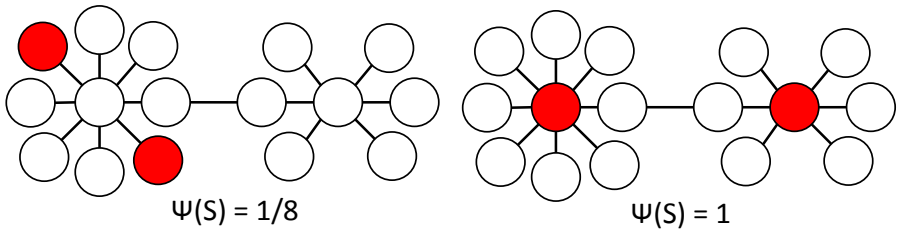

Figure 5.1: A bad (left) and good (right) labeled set according to Ψ

Proof. Let I be the subset of V for which y' is incorrect. If $|I| = 0$ (for example if $L = V$) then the bound holds trivially. We now consider the case where $|I| > 0$. Since y' is consistent with y_L, $I \cap L = \emptyset$. We also know that $\Gamma(I) \neq 0$ since $\Psi(L) > 0$: if $\Gamma(I) = 0$ then I would be a non empty subset of $V \setminus L$ with $\Gamma(I)/|I| = 0$. Thus

$$|I| = \frac{|I|}{\Gamma(I)}\Gamma(I) \leq \frac{1}{\Psi(L)}\Gamma(I)$$

We now argue that $\Gamma(I) \leq \Gamma(V_{y=1}) + \Gamma(V_{y'=1})$, concluding the proof. First note that

$$I = (V_{y=1} \cap V_{y'=0}) \cup (V_{y=0} \cap V_{y'=1})$$

In other words, I is the union of points labeled positive by y but negative by y' and points labeled negative by y but positive by y'. Using the submodularity, non-negativity, symmetry and basic set theory we get

$$\begin{aligned}
\Gamma(I) &\leq \Gamma(V_{y=1} \cap V_{y'=0}) + \Gamma(V_{y=0} \cap V_{y'=1}) \\
&= \Gamma(V_{y=1} \cap V_{y'=0}) + \Gamma(V \setminus (V_{y=1} \cup V_{y'=0})) \\
&= \Gamma(V_{y=1} \cap V_{y'=0}) + \Gamma(V_{y=1} \cup V_{y'=0}) \\
&\leq \Gamma(V_{y=1}) + \Gamma(V_{y'=0}) \\
&= \Gamma(V_{y=1}) + \Gamma(V_{y'=1})
\end{aligned}$$

□

This theorem states that if we make a prediction consistent with the observed labels y_L, we are guaranteed to have a low error rate so long as (1) $\Gamma(V_{y=0})$ is small, (2) $\Gamma(V_{\hat{y}=0})$ is small, and (3) $\Psi(L)$ is large. Perhaps surprisingly, this bound holds no matter how y is chosen. $y_{V \setminus L}$ can in fact even be chosen by an adversary *in response to* $\hat{y}$ (i.e. with full knowledge of our algorithm).

5.3 Active Semi-Supervised Learning Algorithm

Theorem 5.1 immediately implies a semi-supervised prediction method. To minimize the bound we should choose $\hat{y}$ to minimize $\Gamma(V_{\hat{y}=0})$ under the constraint that $\hat{y}_L = y_L$. This prediction method strictly generalizes the min-cut prediction method of [Blum and Chawla, 2001]. Moreover it is possible to compute this prediction in polynomial time, as this is simply a submodular function minimization problem. The error bound also simplifies when we use this prediction method.

Corollary 5.1. *Let Γ be any non-negative, normalized, symmetric submodular function. If y' is chosen to be*

$$y' = \operatorname*{argmin}_{\hat{y} \in \{0,1\}^n : \hat{y}_L = y_L} \Phi(\hat{y})$$

and, $L \subseteq V$, $\Psi(L) > 0$ then

$$||y - y'||^2 \leq 2\frac{\Phi(y)}{\Psi(L)}$$

The error bound also suggests an active learning method: we should choose the labeled set to maximize $\Psi(L)$. However, it is less obvious how to perform this maximization. Note that while $\Psi(L)$ is defined in terms of a submodular function, we will show $\Psi(L)$ is not submodular, so it is not clear that we can use submodular function optimization to maximize $\Psi(L)$. In fact, it is not even immediately clear how to compute $\Psi(L)$.

5.3.1 Computing $\Psi(L)$

We discuss how to compute $\Psi(S)$. Breaking the ratio into a sum

$$\min_{T \subseteq (V \setminus S)} \Big(\Gamma(T) - \lambda |T| \Big) \tag{5.2}$$

gives an expression which can be computed in polynomial time for any fixed S and λ by solving a submodular minimization problem. When Γ is the cut function, this is a mincut problem [Cunningham, 1985]. For a fixed S, it is known there are only $n-1$ many critical values of λ for which the minimizing set $T \subseteq (V \setminus S)$ changes [Fujishige, 2005]. We also have the following result taken from Fujishige [2005]

Theorem 5.2 (Fujishige [2005])**.** *Assume* $g(T) \geq 0$*,* $h(T) \geq 0$*,* $g(\emptyset) = 0$*,* $h(\emptyset) = 0$*, and* $h(T) \neq 0$ *for some* T*.* $\lambda^* = \min_{T:h(T)\neq 0} \frac{g(T)}{h(T)}$ *iff*

$$\forall \lambda \leq \lambda^*, \quad \min_T \big(g(T) - \lambda h(T)\big) = 0$$

and

$$\forall \lambda > \lambda^*, \quad \min_T \big(g(T) - \lambda h(T)\big) < 0$$

This theorem motivates a method for computing $\Psi(S)$ for any $S \subset V$: if we can find the largest critical value λ^* which makes

$$\min_{T \subseteq (V \setminus S)} \Big(\Gamma(T) - \lambda |T|\Big) = 0. \tag{5.3}$$

and a *non-empty* T such that $\Gamma(T) - \lambda^* |T| = 0$ we are done. Algorithm 5.1 shows an iterative algorithm which searches for λ^* and T [Cunningham, 1985]. This method starts with λ set to an overestimate of $\Psi(S)$ (for example $\Gamma(V \setminus S)/|V \setminus S|$) and then iteratively computes a decreased upper estimate stopping when (5.3) holds. Cunningham [1985] prove this converges after $O(n)$ iterations.

Given this method we can also compute $\Psi(V)$. Recall that $\Psi(V)$ was defined as a special case to be $\max_{S \subset V} \Psi(S)$. Since $\Psi(S)$ is monotonically increasing, $\Psi(V) = \max_{s \in V} \Psi(V \setminus \{s\})$, and we can compute $\Psi(V)$ using $|V|$ evaluations of $\Psi(S)$ for $S \subset V$.

5.3.2 Maximizing $\Psi(L)$

We now consider the problem of choosing a labeled set to minimize the error bound. There are in fact two related problems. One is to find a minimal size S such that $\Psi(S) \geq \lambda$ for

Algorithm 5.1 Compute $\Psi(S)$

$T' \leftarrow V \setminus S$

repeat

$T \leftarrow T'$

$\lambda \leftarrow \frac{\Gamma(T)}{|T|}$

$T' \leftarrow \underset{\hat{T} \subseteq (V \setminus S)}{\operatorname{argmin}} \left(\Gamma(\hat{T}) - \lambda |\hat{T}| \right)$

until $\Gamma(T') - \lambda |T'| = 0$

return $\frac{\Gamma(T)}{|T|}$

some target λ.

$$S^* = \underset{S:\Psi(S) \geq \lambda}{\operatorname{argmin}} |S| \tag{5.4}$$

We also consider constrained maximization

$$S^* = \underset{S:|S| \leq k}{\operatorname{argmax}} \Psi(S) \tag{5.5}$$

These two problems correspond to picking the smallest labeled set achieving a target error bound and picking the labeled set of size k achieving the best error bound. We've shown $\Psi(S)$ can be computed in polynomial time, but it is not immediately obvious how to solve (5.4) or (5.5).

If $\Psi(S)$ were a submodular function, these two problems would correspond to submodular set cover and cardinality constrained submodular maximization. In this case we could apply the standard greedy algorithm. Unfortunately one can show $\Psi(S)$ is *not* submodular or in fact supermodular.

Lemma 5.1. Ψ *is not submodular.*

Proof. Consider a binary weight graph consisting of a cycle of length 4 with vertices 1, 2, 3, 4 along the cycle. For this graph

$$\Psi(\emptyset) = 0 \qquad \Psi(\{1\}) = \frac{2}{3}$$

$$\Psi(\{3\}) = \frac{2}{3} \qquad \Psi(\{1,3\}) = 2$$

So,

$$\Psi(\{1,3\}) - \Psi(\{3\}) = \frac{4}{3} > \Psi(\{1\}) - \Psi(\emptyset) = \frac{2}{3}$$

□

Lemma 5.2. *Ψ is not supermodular.*

Proof. Consider a binary weight graph consisting of a chain of length 4 with vertices 1, 2, 3, 4 along the chain. For this graph

$$\Psi(\emptyset) = 0 \qquad \Psi(\{2\}) = \frac{1}{2}$$
$$\Psi(\{1\}) = \frac{1}{3} \qquad \Psi(\{1,2\}) = \frac{1}{2}$$

So,

$$\Psi(\{1,2\}) - \Psi(\{2\}) = 0 < \Psi(\{1\}) - \Psi(\emptyset) = \frac{1}{3}$$

□

To maximize $\Psi(S)$, we consider (5.2) as a function of S

$$F_\lambda(S) \triangleq \min_{T \subseteq (V \setminus S)} \Big(\Gamma(T) - \lambda |T| \Big)$$

We first observe that sets which satisfy the constraint $\Psi(S) \geq \lambda$ are exactly the sets which satisfy $F_\lambda(S) = 0$. This result is largely implicit in previous work (e.g. Theorem 5.2), but it is useful to restate it in terms of functions of S as this is crucial to our approach.

Lemma 5.3. *For any $\lambda \in [0, \Psi(V)]$, S, $F_\lambda(S) = 0$ iff $\Psi(S) \geq \lambda$*

Proof. If $S = V$ then $F_\lambda(S) = \Gamma(\emptyset) = 0$ and $\Psi(S) \geq \lambda$ by definition. We therefore restrict our attention to $S \subset V$.

We first show if for some $S \subset V$ $F_\lambda(S) = 0$ then $\Psi(S) \geq \lambda$. Assume for the sake of contradiction that $F_\lambda(S) = 0$ and $\Psi(S) < \lambda$. Therefore there is some $T \subseteq (V \setminus S) : T \neq \emptyset$ with $\frac{\Gamma(T)}{|T|} < \lambda$. Rearranging terms we have $\Gamma(T) - \lambda |T| < 0$ which contradicts $F_\lambda(S) = 0$.

We now show if for some $S \subset V$ $\Psi(S) \geq \lambda$ then $F_\lambda(S) = 0$. Assume for the sake of contradiction that $\Psi(S) \geq \lambda$ and $F_\lambda(S) < 0$. Therefore there is some $T \subseteq (V \setminus S)$ with

$T \neq \emptyset$ and $\Gamma(T) - \lambda|T| < 0$. Again rearranging terms we get $\frac{\Gamma(T)}{|T|} < \lambda$ which contradicts $\Psi(S) \geq \lambda$. □

Note that we can evaluate $F_\lambda(S)$ in polynomial time through submodular function minimization. We can also show that for any fixed λ, F_λ is monotone and submodular. The fact that F_λ is submodular is a special case of the result that the convolution of a submodular function with a modular function is submodular [Narayanan, 1997]. See also Proposition 2 of Cunningham [1985].

Lemma 5.4. *For any λ, $F_\lambda(S)$ is submodular and monotone non-decreasing*

Proof. Monotonicity is obvious: adding elements to S decreases the domain over which the minimization occurs, so F_λ increases. A function F is submodular iff $F(V \backslash S)$ is submodular, so it suffices to show that

$$F'_\lambda(S) = F_\lambda(V \setminus S) = \min_{T \subseteq S} \Big(\Gamma(T) - \lambda|T| \Big)$$

is submodular. Narayanan [1997] defines the lower convolution of G and H to be

$$G * H(S) = \min_{T \subseteq S} \Big(G(T) + H(S \setminus T) \Big)$$

Narayanan [1997] show that if G is submodular and H is modular then $G * H$ is submodular. We see that F'_λ is the lower convolution of $G(T) \triangleq \Gamma(T) - \lambda|T|$, which is submodular, with $H(T) \triangleq 0$, which is modular. □

These two lemmas combine to give an approximation algorithm for (5.4) (see Algorithm 5.2).

Theorem 5.3. *Assume $\lambda \in [0, \Psi(V)]$ is an integer and Γ is integer valued. Applying the greedy algorithm to F_λ until $F_\lambda(S) \geq 0$ gives a set $S \subseteq V$ with $\Psi(S) \geq \lambda$ and*

$$|S| \leq (1 + \ln \lambda n) \min_{S^* : \Psi(S^*) \geq \lambda} |S^*|$$

Proof. From Lemma 5.3, finding such a set with minimum size is equivalent to solving

$$\operatorname*{argmin}_{S : F_\lambda(S) \geq 0} |S|$$

Algorithm 5.2 Minimize $|S|$ s.t. $\Psi(S) \geq \lambda$

Define $F_\lambda(S) \triangleq \min_{T \subseteq (V \setminus S)} \Gamma(T) - \lambda|T|$

// Greedily maximize $F_\lambda(S)$

$S \leftarrow \emptyset$

while $F_\lambda(S) < 0$ **do**

$s \leftarrow \underset{\hat{s} \in V}{\operatorname{argmax}} F_\lambda(S \cup \{\hat{s}\}) - F(S)$

$S \leftarrow S \cup \{s\}$

end while

From Lemma 5.4, F_λ is submodular so this is a submodular set cover problem. Wolsey [1982] shows that if F is integer, monotone, submodular, and normalized then the greedy algorithm gives a $1+\ln F(V)$ approximation for submodular set cover. $F_\lambda(S)$ is submodular, monotone, and integer valued. $F_\lambda(S)$ can be normalized by adding λn, giving the result. □

This theorem was inspired by a remark in the introduction of a recent paper of Gupta et al. [2010a]. The authors mention how submodular function maximization can be applied to give an approximation for a max-min version of mincut. Our problem is a max-min optimization over $\Gamma(T)/|T|$ instead of $\Gamma(T)$.

For non-integral but rational λ we can rescale Γ and λ in order to make λ an integer. For general Γ or λ the result of Wolsey [1982] would add an extra normalization term inside the log that is equal to the inverse of the smallest non-zero gain of F_λ.

In the worst case, the algorithm evaluates $F_\lambda(S)$ n^2 times, making the run time of the algorithm n^2 that of the submodular function minimization method used. Submodular function minimization can be solved in polynomial time, but currently the fastest known methods in theory require $O(n^6)$ time [Iwata and Orlin, 2009]. The minimum norm algorithm of Fujishige et al. [2006] is generally regarded as the fastest known algorithm in practice, but its worst case running time is unknown. If Γ is the graph cut function we can use mincut in place of submodular function minimization, greatly improving performance. Standard techniques [Minoux, 1978] can also be used to greatly reduce the number of function evaluations needed. Empirically, these methods often reduce the number of function

evaluations to $O(n)$.

We can also use Theorem 5.3 to give a bicriterion guarantee for the maximization problem (5.5).

Lemma 5.5. *Assume k is a postive integer and Γ is integer valued. There is a pseudo-polynomial time algorithm finding a set S with $\Psi(S) \geq \max_{S^*:|S^*|\leq k} \Psi(S^*)$ and*

$$|S| \leq (1 + 2 \ln n\Psi(S))k$$

Proof. Define $\lambda^* \triangleq \max_{S^*:|S^*|\leq k} \Psi(S^*)$. First note that for any S, $\lambda = \Psi(S)$ is a rational number and we can scale λ and Γ by at most n to make them both integer valued. Say that we have such a rational number $0 \leq \lambda \leq \Psi(V)$. Let S be the result of scaling λ and Γ to an integer then applying the greedy algorithm to F_λ until $F_\lambda(S) \geq 0$. If $\lambda \leq \lambda^*$ then from Theorem 5.3 we must have $|S| \leq (1+2\ln n\lambda)k$. Conversely if $|S| > (1+2\ln n\lambda)k$ then $\lambda > \lambda^*$. Here the extra factor of 2 is due to scaling, which adds an extra factor of at most n inside of the log.

There are at most $n^2\Psi(V)$ valid rational values between 0 and $\Psi(V)$, so we can enumerate them. The algorithm then performs binary search to find a value of λ where the greedy algorithm gives $|S| \leq (1 + 2\ln n\lambda)k$ and for which the next largest value of λ results in $|S| > (1 + 2\ln n\lambda)k$. □

5.4 Negative Results

We have given an approximation algorithm for finding a minimum size set satisfying $\Psi(S) \geq \lambda$ and a bicriteria approximation for maximizing $\Psi(S)$. These results raise the question: can we do better? We give a partial answer to this question: we show that there are no exact algorithms for either of these problems and no PTAS (polynomial time approximation scheme) for the cost minimization version (5.4). A problem has a PTAS if there is a constant factor approximation algorithm for every constant. Our results do not exclude the possibility of a constant factor approximation, however.

Theorem 5.4. *Assume $\Gamma(S)$ is graph cut. Given as input W, k, and λ, determining whether there is a set S with $|S| \leq k$ and $\Psi(S) \geq \lambda$ is NP-complete.*

Proof. We show that, assuming W is a binary weight, cubic graph, there is a set $S \subset V$ with $|S| \leq k$ and $\Psi(S) \geq 3$ iff the graph has a vertex cover of size k. Recall that a cubic graph is a graph where every node has degree 3 and a vertex cover is a set $S \subseteq V$ with either $i \in S$ or $j \in S$ for every $W_{i,j} \neq 0$. The result then follows from the fact that determining if a graph has a vertex cover of size k is NP-complete, even when the graph is restricted to cubic graphs [Garey and Johnson, 1979]. Note that in a cubic graph a set S is a vertex cover iff

$$\Gamma(V \setminus S) = 3|V \setminus S| \tag{5.6}$$

This is because every vertex not in S must have exactly three neighbors in S.

We show that $\min_{T \subseteq (V \setminus S)} \Big(\Gamma(T) - 3|T| \Big) = 0$ iff $\Gamma(V \setminus S) - 3|V \setminus S| = 0$. The result then follows from Lemma 5.3. Let T be any set. Consider any $s \in (V \setminus S)$ with $s \notin T$. If we add s to T we increase $\Gamma(T)$ by at most 3 but we increase $|T|$ by 1 so

$$\Gamma(T + s) - 3|T + s| \leq \Gamma(T) - 3|T| = 0$$

Applying this recursively we therefore have

$$\Gamma(V \setminus S) - 3|V \setminus S| \leq \Gamma(T) - 3|T|$$

Because T was any set, we therefore have

$$\Gamma(V \setminus S) - 3|V \setminus S| = \min_{T \subseteq (V \setminus S)} \Big(\Gamma(T) - 3|T| \Big)$$

□

Note we've shown the result for Γ equal to the cut function, but clearly the problem with arbitrary Γ is only harder. Since the decision problem is NP-complete, both (5.4) and (5.5) are also NP-complete, as polynomial time algorithms for these problems would imply polynomial time algorithms for the decision problem. The reduction used in the proof of Theorem 5.4 is approximation preserving (specifically it leaves the problem size unchanged), and computing minimum vertex cover on a cubic graph is known to be APX-complete [Alimonti and Kann, 1997], so we also get the following corollary.

Corollary 5.2. *There is no PTAS for (5.4) unless P=NP.*

Theorem 5.4 shows that our proposed algorithm is approximately optimal in the sense that (unless P=NP) no polynomial time algorithm can exactly minimize the error bound which our algorithm approximately minimizes. A different question is whether or not the error bound itself (Theorem 5.1) is optimal. We show no other error bound of the same form can be strictly better.

Theorem 5.5. *Assume* $\Gamma(S)$ *is symmetric and submodular and* $\Gamma(V) = 0$. *Assume that for some non-negative function* $G(S)$ *depending only on* Γ *and* S *we have*

$$||y - y'||^2 \leq \frac{1}{G(L)}(\Phi(y) + \Phi(y'))$$

for any L, y, *and* y' *consistent with* y_L. *We must have* $G(L) \leq \Psi(L)$ *for every* L *with* $\Psi(L) \neq 0$.

Proof. Assume for some such G we have $G(L) > \Psi(L)$ for some L with $\Psi(L) \neq 0$. Set y' to be a vector of all ones. Note $\Phi(y') = 0$ in this case. Let

$$T = \underset{\hat{T} \subseteq (V \setminus L): \hat{T} \neq \emptyset}{\operatorname{argmin}} \frac{\Gamma(\hat{T})}{|\hat{T}|}$$

Set $y_i = 1$ for $i \in V \setminus T$ and $y_i = -1$ for $i \in T$. By construction we have

$$\Psi(L) = \frac{\Gamma(T)}{|T|} = \frac{\Phi(y) + \Phi(y')}{||y - y'||^2}$$

We then have $G(L) > (\Phi(y)+\Phi(y'))/||y-y'||^2$ which contradicts our assumption $||y-y'||^2 \leq (\Phi(y) + \Phi(y'))/G(L)$ □

$\Psi(L) \neq 0$ holds under mild assumptions. For example, for the cut function Γ, $\Psi(L) > 0$ when L contains at least one node in each connected component of W.

5.5 Normalized Error Bound

In this section we consider a different error bound and algorithm that replaces the Ψ function with a normalized term. Recall that the normalized cut value for a set $S \subset V$ is

$$\frac{\Gamma(S)}{\min(|S|, |V \setminus S|)} \tag{5.7}$$

where $\Gamma(S)$ is graph cut. In other words, normalized cut is the ratio between the cut value for S and minimum of the size of S and its complement. Computing the minimum normalized cut for a graph is NP-hard. In this section we will again consider the more general case where we have an arbitrary symmetric submodular function (not necessarily graph cut).

For any symmetric submodular function $\Gamma(S)$ and set T define

$$\Gamma_T(S) \triangleq \frac{1}{2}(\Gamma(S \cap T) + \Gamma(T \setminus S) - \Gamma(T))$$

When Γ is the cut function $\Gamma_T(S)$ is the sum of the edges crossing between $S \cap T$ and $T \setminus S$ (the cut of S within the subgraph induced by T).

Proposition 5.1. *If $\Gamma(S)$ is symmetric and submodular then for any T $\Gamma_T(S)$ is symmetric and submodular.*

Now define

$$\Xi(S) = \min_{T \subset S} \frac{\Gamma_S(T)}{\min(|T|, |S \setminus T|)}$$

This essentially measures the density of a set: it is the normalized cut "within" S.

Consider the following method: 1) partition the set of nodes V into clusters $S_1, S_2, ...S_k$, 2) for each cluster request sufficient labels to determine the majority class for that cluster, and 3) label all nodes in each cluster with the majority label for the cluster. We show that when each of S_i is dense (according to $\Xi(S_i)$), this method is guaranteed to have low error.

Theorem 5.6. *Let $\Gamma(S)$ be a non-negative symmetric submodular function and $S_1, S_2, ...S_k$ be a partition of V. If $\min_i \Xi(S_i) > 0$ and $\hat{y}$ labels every $v \in S_i$ according to the majority label for S_i then*

$$||y - \hat{y}||^2 \leq \sum_i \frac{\Gamma_{S_i}(V_{y=0})}{\Xi(S_i)}$$

Proof. Let I be the set of incorrectly labeled nodes (errors). We consider the intersection of I with each of the clusters. Let $I_i \triangleq |I \cap S_i|$. $I = \bigcup_{i=1}^k I_i$ Note that $|I_i| \leq |S_i \setminus I_i|$ since we labeled cluster according to the majority label for the cluster. We show that for every

non empty I_i, $|I_i| < \frac{\Gamma_{S_i}(V_{y=0})}{\Xi(S_i)}$. The bound then follows. Note that for any non empty I_i we must have $\Gamma_{S_i}(I_i) > 0$ since by assumption $\Xi(S_i) > 0$. Therefore

$$\begin{aligned}
|I_i| &= \Gamma_{S_i}(I_i)\frac{|I_i|}{\Gamma_{S_i}(I_i)} \\
&= \Gamma_{S_i}(I_i)\frac{\min(|I_i|, |S_i \setminus I_i|)}{\Gamma_{S_i}(I_i)} \\
&\leq \frac{\Gamma_{S_i}(I_i)}{\Xi(S_i)}
\end{aligned}$$

The proof then follows exactly that of Theoerm 5.1 to show $\Gamma_{S_i}(I_i) \leq \Gamma_{S_i}(V_{y=0}) + \Gamma_{S_i}(V_{\hat{y}=0})$. Because all of S_i has the same label $\Gamma_{S_i}(V_{\hat{y}=0}) = 0$ and the bound follows. □

In this bound, $\Xi(S_i)$ is a measure of the density of a cluster. Computing $\Xi(S_i)$ for a particular cluster is NP-hard, but for the case where $\Gamma(S)$ is graph cut there are approximation algorithms. However, even in this case we are not aware of approximation algorithms for computing a partition such that $\min_i \Xi(S_i)$ is minimized. This is different from the standard normalized cut clustering problem; we do not care if clusters are strongly connected to each other only that each cluster is internally dense. We have tried this method using standard normalized cut clustering algorithms and found this to give good real world performance. We report some of these results in the following section. However, it remains an interesting open question to design a clustering algorithm for directly maximizing $\min_i \Xi(S_i)$. An approach we have not yet tried is to use the error bound to choose between the results of different clustering algorithms.

Note in practice we will likely not know the majority labels of S_i but rather some estimate. We now consider the problem of estimating the majority class for a cluster. If we uniformly sample labels from a cluster, standard results give that the probability of incorrectly estimating the majority decreases exponentially with the number of labels if the fraction of nodes in the minority class is bounded away from $1/2$ by a constant. We now show that if the labels are sufficiently smooth and the cluster is sufficiently dense then the fraction of nodes in the minority class is small.

Theorem 5.7. *Assume $\Xi(S) > 0$. The fraction of nodes in the minority class of S is at*

most

$$\frac{\Gamma_S(V_{y=0})}{\Xi(S)|S|}$$

Proof. Let S^- be the subset of S belonging to the minority class and S^+ be the subset belonging to the other class. Let f be the fraction of nodes in the minority class. If $|S^-| = 0$ then the bound holds trivially. We therefore assume $|S^-| > 0$. In this case $\Gamma_S(V_{y=0}) = \Gamma_S(V_{y=1}) > 0$ since by assumption $\Xi(S) > 0$. We now have

$$\begin{aligned} f &= \frac{|S^-|}{|S|} = \frac{|S^-|}{|S|} \frac{\Gamma_S(V_{y=0})}{\min(|S^+|, |S^-|)} \frac{\min(|S^+|, |S^-|)}{\Gamma_S(V_{y=0})} \\ &= \frac{\Gamma_S(V_{y=0})}{|S|} \frac{\min(|S^+|, |S^-|)}{\Gamma_S(S^+)} \\ &\leq \frac{\Gamma_S(V_{y=0})}{\Xi(S)|S|} \end{aligned}$$

□

If we have an estimate of $\Gamma_{S_i}(V_{y=0})$, we can use this bound and an approximation of $\Xi(S_i)$ to determine the number of labels needed to estimate the majority class with high confidence. In our experiments, we simply request a single label per cluster.

5.6 Experiments

We tested our proposed methods on both a set of benchmark graph-based learning problems and on a movie recommendation problem over a hypergraph.

5.6.1 Learning on Graphs

We tested our method on a set of benchmark data sets from Chapelle et al. [2006] (also used in previous work [Guillory and Bilmes, 2009]) and two citation graph data sets, Citeseer and Cora, from Sen et al. [2008] (Cora was also used by Cesa-Bianchi et al. [2010a]). In these experiments we set Γ to be standard graph cut. We tried our method for maximizing $\Psi(L)$ (Lemma 5.5) in conjunction with minimum cut prediction [Blum and Chawla, 2001] and also the version of label propagation proposed by Bengio et al. [2006] (using $\mu = 10^{-6}$, $\epsilon = 10^{-6}$, and class mass normalization). Minimum cut prediction is more directly motivated by our theory (when Γ is graph cut) but label propagation sometimes works better in practice. We

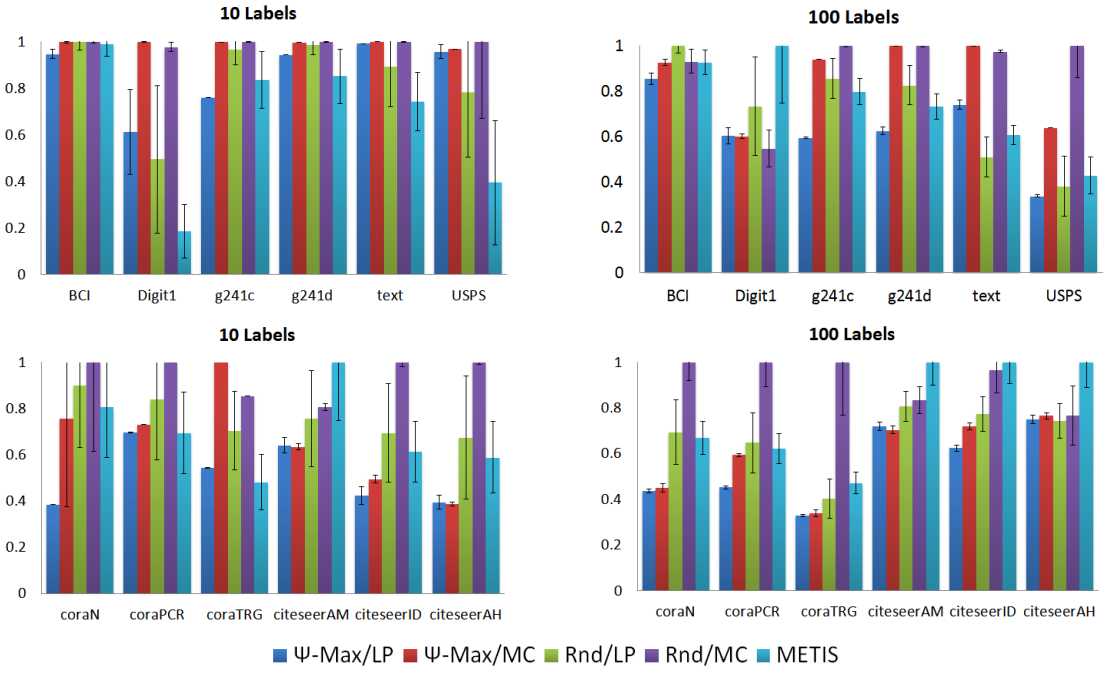

Figure 5.2: Relative error.

note Theorem 5.1 still holds for label propagation prediction if we set L to be the set of points on which our predictions agree with the observed labels. Also, in our experiments we use binary search to find a set S with $|S| \leq k$ (as opposed to $|S| < \tilde{O}(k)$), and we do not perform binary search over all rational values of $\Psi(S)$ but instead stop when the relative difference between the upper and lower bounds is less than .0001.

We compare against three baselines: minimum cut prediction with a random labeled set, label propagation prediction with a random labeled set, and the METIS active learning heuristic [Guillory and Bilmes, 2009]. For predicting with k labeled points, the METIS heuristic partitions the graph into k parts using the METIS graph partitioning program [Karypis and Kumar, 1999], requests a label for a single random point in each of the k parts, and then labels each part according to that label. This method was found to outperform a variety of other heuristic methods on the benchmark data sets. The METIS heuristic is inspired by Theorem 5.6 but has no theoretical guarantees. We average over 1000 runs of the baseline methods for each dataset / label count. For our methods we average over 100 different runs of the final selection(after binary search for λ^*) on different random permutations of the data.

For the benchmark data sets we construct a k-nearest neighbor graph with $k = 10$. These data sets are of size $n = 1500$ except for BCI which is of size $n = 400$. We use an unweighted graph for all methods except label propagation for which we used a Gaussian kernel weighted graph ($W_{i,j} = e^{||x_i - x_j||^2/2\sigma^2}$). We set σ with a heuristic from Chapelle et al. [2006]: we use $1/3$ the average distance to the k-th nearest neighbor. We chose to use an unweighted graph for our method as in some preliminary experiments on other data sets we found it can be sensitive to the choice of σ.

On the citation graph data sets, following the setup used by Cesa-Bianchi et al. [2010a], we use the largest connected component of the graph, group together small classes to form more balanced classes, and perform one-vs-rest binary classification. We do not use the feature vectors for the documents on these data sets–only the citation graph structure. The edges in these data sets are unweighted, and we treat them as undirected. The Cora subset we use is of size $n = 2485$, and the Citeseer subset is of size $n = 2110$. The class groupings we use for Cora are Neural Networks (coraN), Theory / Reinforcement Learning

/ Genetic Algorithms (coraTRG), and Probabilistic Methods / Case Based / Rule Learning (coraPCR). For Citeseer we use AI / ML (citeseerAM), IR / DB (citeseerID), and Agents / HCI (citeseerAH).

Figure 5.6.1 plots relative error compared to the worst method for different data set / label counts. The top row is k-nn graphs, bottom row citation graphs. Ψ-max is our method, Rnd is random selection, MC is minimum cut, LP is label propagation, and METIS is the clustering heuristic. Bars show one standard deviation. On the benchmark data sets, our method combined with label propagation performs better than the others on 6/12 data set / label combinations, but on some problems our method hurts. This is not entirely surprising in light of previous observations made about minimum cut prediction and k-nearest neighbors graphs [Blum et al., 2004]. We speculate a different choice of graph construction or Γ is needed to get more consistent improvement. Graph construction and hyper parameter selection for semi-supervised learning is in general a difficult problem.

However, on the two citation graph data sets where the graph is not constructed by us (i.e. the graph is part of the data), our method has a significant and more consistent benefit. On 9/12 of the classification tasks one of our active learning methods performs best (either the minimum cut or label propagation version), and the difference between our active learning methods and the other methods is often large. On all but one of the classification problems the label propagation version of our method is within 1 percent of the best method. The different variations of Cora (Citeseer) differ only in their labels, so our method selects the same labeled points for each of these variations. This is a feature of our batch setting. Results also confirm our method reduces variance in error. We think our method's strong performance on these natural graph data sets suggests that, when the choice of Γ is appropriate for the data, our method performs well.

In a followup experiment we tried running METIS 100 times per trial instead of once, keeping the partition with the lowest cut value. This variation helped some on a few data set / label combinations (the biggest was a 10% relative decrease in error on Digit1 / 10 where METIS already performed well) but overall had very little effect (relative to variance in error) and didn't change trends in the results. We have also begun experimenting with alternative graph constructions and objectives for the benchmark data sets, for example the

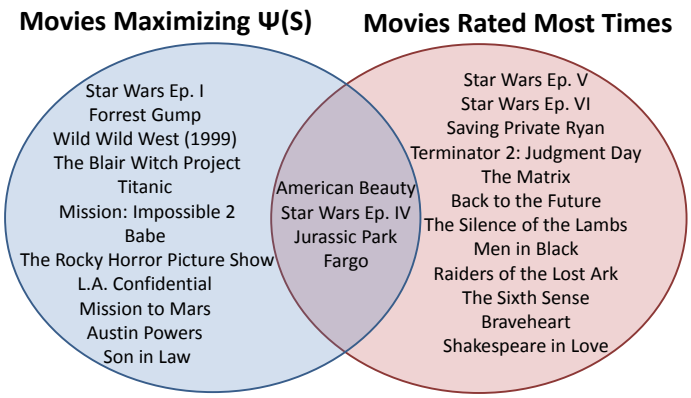

Figure 5.3: Movies informative about taste.

δ construction of Blum et al. [2004], but have not yet achieved consistent improvement

5.6.2 Choosing Movies to Rate

In this section we discuss a problem setting which uses the added generality of Corollary 5.1. We consider the problem of training a collaborative filtering system for a new user: which items (e.g. movies) should we ask a new user to evaluate first so that we can then give the user accurate recommendations?

We treat this problem as an active learning problem over a hypergraph by constructing a hypergraph in which nodes are items and edges encode user preferences. We make for each user an edge connecting all movies that the user rated higher than 3 (out of 5) stars and an edge connecting all movies that the user rated lower than 3 stars. With a graph constructed like this, a partition of the hypergraph which cuts few hypergraph edges corresponds to a grouping of movies that is on average consistent with all users' preferences: on average, movies that are liked by the same user will be on the same side of the cut as will movies that are disliked by the same user. More complicated methods could assign different weights to the hypergraph edges or create more than two edges per user.

We set $\Gamma(S)$ to be the hypergraph cut function counting the number of hypergraph

edges crossing S and $V \setminus S$. Corollary 5.1 then suggests a semi-supervised learning method for predicting which items a user likes based on their likes and dislikes for a subset of items L. This method guarantees low error so long as the user's true preferences y are mostly consistent with previously seen ratings. This Corollary also suggests that in order to minimize error a new user should rate a set of items L which maximizes $\Psi(L)$.

We tested our Ψ maximization method on the 1 Million Ratings version of the MovieLens data. We use as a ground set all movies which received more than 10 ratings (3233 nodes). The construction method described above then gives 11479 hypergraph edges (about 737000 user/movie connections). We compute hypergraph mincuts using the flow method of Yang and Wong [1996]. Figure 5.3 shows the 16 movies selected by our method in order to minimize $|S|$ subject to $\Psi(S) \geq 2.5$. We compare to the 16 movies which were rated the most number of times.

There is some overlap between the two lists: movies that are rated many times will have high degree in the hypergraph, and therefore may be useful for learning. However, the movies that are selected by our method are more diverse in certain ways and therefore potentially more useful for learning. For example, our method chose movies that are controversial (The Blair Witch Project which was liked/disliked with proportion about 1.18:1 in the data set), a movie considered a cult classic (The Rocky Horror Picture Show), a children's movie (Babe), and even unpopular movies (Son in Law, which received an average rating of 2.67/5). Unpopular movies may be valuable as they can confirm which movies a user does not like. In comparison, the movies rated most often are largely popular movies.

Chapter 6

ONLINE SUBMODULAR SET COVER

The problems we have considered so far are all *single-round* active learning problems where the learning algorithm solves a single problem in isolation. An alternative setting is *repeated* active learning where the learning algorithm solves a series of related active learning problems. For example, consider a medical diagnosis problem where at reach round we must choose a sequence of tests to perform on a patient with an unknown illness. In such a setting there may be correlations between the unknown illnesses of successive patients, and ideally our learning algorithm should take advantage of these correlations when they are present.

Repeated active learning is related to online ranking. In an online ranking problem, at each round we choose an ordered list of items and then incur some loss. Problems with this structure include search result ranking, ranking news articles, and ranking advertisements. In search result ranking, each round corresponds to a search query and the items correspond to search results. When the list of items is a sequence of questions to ask or tests to perform, online ranking is a kind of repeated active learning. In particular, a list of questions specifies a very simple active learning strategy.

In many online ranking problems the loss incurred at each round is the number of items in the list needed to achieve some goal. For example, in search result ranking a reasonable loss is the number of results the user needs to view before they find the complete information they need. Similarly, in a repeated diagnosis problem a good loss is the number of tests needed before we can make a confident diagnosis. We propose an approach to problems like these using a new online prediction version of submodular set cover, which we simply call *online submodular set cover*.

6.1 Problem

At each time step $t = 1 \dots T$ we choose a sequence of elements $S^t = (v_1^t, v_2^t, \dots v_n^t)$ where each v_i^t is chosen from a ground set V of size n (we use a superscript for rounds of the online problem and a subscript for other indices). After choosing S^t, an adversary reveals a submodular, monotone, normalized function F^t, and we suffer loss $\ell(F^t, S^t)$ where

$$\ell(F^t, S^t) \triangleq \min\big(\{n\} \cup \{i : F^t(S_i^t) \geq 1\}_i\big) \tag{6.1}$$

and $S_i^t \triangleq \bigcup_{j \leq i} \{v_j^t\}$ is defined to be the set containing the first i elements of S^t (let $S_0^t \triangleq \emptyset$). Note ℓ can be equivalently written $\ell(F^t, S^t) \triangleq \sum_{i=0}^{n} I(F^t(S_i^t) < 1)$ where I is the indicator function. Intuitively, $\ell(F^t, S^t)$ corresponds to a bounded version of cover time: it is the number of items up to n needed to achieve $F^t(S) \geq 1$ when we select items in the order specified by S^t. Thus, if coverage is not achieved, we suffer a loss of n. We assume that $F^t(V) \geq 1$ (therefore coverage is achieved if S^t does not contain duplicates) and that the sequence of functions $(F^t)_t$ is chosen in advance (by an oblivious adversary). The goal of our learning algorithm is to minimize the total loss $\sum_t \ell(F^t, S^t)$.

To make the problem clear, we present it first in its simplest, full information version. However, we will later consider more complex variations including (1) a version where we only produce a list of length $k \leq n$ instead of n, (2) a multiple objective version where a set of objectives $F_1^t, F_2^t, \dots F_m^t$ is revealed each round, (3) a bandit (partial information) version where we do not get full access to F^t and instead only observe $F^t(S_1^t), F^t(S_2^t), \dots F^t(S_n^t)$, and (4) a contextual bandit version where there is some context associated with each round.

We argue that online submodular set cover, as we have defined it, is an interesting and useful model for ranking and repeated active learning problems. In a search result ranking problem, after presenting search results to a user we can obtain implicit feedback from this user (e.g., clicks, time spent viewing each result) to determine which results were actually relevant. We can then construct an objective $F^t(S)$ such that $F^t(S) \geq 1$ iff S covers or summarizes the relevant results. Alternatively, we can avoid explicitly constructing an objective by considering the bandit version of the problem where we only observe the values $F^t(S_i^t)$. For example, if the user clicked on k total results then we can let $F(S_i^t) \triangleq c_i/k$

where $c_i \leq k$ is the number of results in the subset S_i which were clicked. Note that the user may click an arbitrary set of results in an arbitrary order, and the user's decision whether or not to click a result may depend on previously viewed and clicked results. All that we assume is that there is some unknown submodular function explaining the click counts. If the user clicks on a small number of very early results, then coverage is achieved quickly and the ordering is desirable. This coverage objective makes sense if we assume that the set of results the user clicked are of roughly equal importance and together summarize the results of interest to the user.

In the medical diagnosis application, we can define $F^t(S)$ to be proportional to the number of candidate diseases which are eliminated after performing the set of tests S on patient t. If we assume the result of a test is a fixed function of the patient's disease and that each test result eliminates a fixed set of candidate diseases, this objective is submodular and can be inferred after the end of the round. More formally, let H be the set of all candidate diseases, and let $h(S)$ be the set of of test results we would observe after performing the tests S on a patient with disease $h \in H$. Define $V(h(S))$ to be the subset of H consistent with the test results $h(S)$. Define

$$F^t(S) = \frac{|H \setminus V(h^t(S))|}{|H| - 1}$$

where h^t is the tth patient's disease. This is just a variation of the version space reduction objective of Section 2.7. Note that, as in the search result ranking problem, F^t is not initially known but is known after we have chosen S^t and suffered loss $\ell(F^t, S^t)$ (after we have identified h^t). We can relax the assumption that h^t is always identified and test results are a fixed function of the patient's disease if we consider the bandit setting where only the values $F^t(S_i^t)$ are observed. We can also relax the assumption that each test result eliminates a fixed set of diseases by moving to a noisy setting. For this we can use the objectives in Section 4.3.

Because we select a fixed sequence of questions at the start of each round, the active learning policies produced by our approach are in some sense not interactive: within a round, responses to questions do not affect the order in which questions are asked. However, between rounds the algorithm is interactive: the responses to questions asked on previous

rounds can affect the order of the questions on the current round. This kind of algorithm makes sense for certain applications. For example, consider choosing a ranking of questions for a web based survey. If we display all the questions at once in a single list, it makes sense to require that the algorithm produces a fixed ranking at the start of a round. For other applications, requiring a fixed sequence of questions within each round is a somewhat unrealistic simplifying assumption. Developing an approach which is interactive both within and between rounds is an open problem.

6.2 Related Problems

Recall that that ranking with submodular valuations (see Section 1.3.2) is the problem of selecting a permutation of V which approximately minimizes the average cover time of a weighted set of submodular, monotone objectives. The offline approximation algorithm for ranking with submodular valuations Azar and Gamzu [2011] will be a crucial tool in our analysis of online submodular set cover. In particular, this offline algorithm can viewed as constructing a sequence which approximately minimizes empirical loss (e.g. the best single permutation for a sequence of objectives *in hindsight* after all the objectives are known). Recently the ranking with submodular valuations problem was extended to metric costs [Im and Nagarajan, 2011]. Im and Nagarajan [2011] also extend the problem to a stochastic setting where the ground set is a set of independent random variables whose realizations are revealed only choosing an element.

Online learning is a well-studied problem [Cesa-Bianchi and Lugosi, 2006]. In one standard setting, the online learning algorithm has a collection of actions $\mathcal{A}$, and at each time step t the algorithm picks an action $S^t \in \mathcal{A}$. The learning algorithm then receives a loss function ℓ^t, and the algorithm incurs the loss value for the action it chose $\ell^t(S^t)$. We assume $\ell^t(S^t) \in [0, 1]$ but make no other assumptions about the form of loss. The performance of an online learning algorithm is often measured in terms of *regret*, the difference between the loss incurred by the algorithm and the loss of the best single fixed action in hindsight: $R = \sum_{t=1}^{T} \ell^t(S^t) - \min_{S \in \mathcal{A}} \sum_{t=1}^{T} \ell^t(S)$. There are randomized algorithms which guarantee $\mathbb{E}[R] \leq \sqrt{T \ln |\mathcal{A}|}$ for *adversarial* sequences of loss functions [Freund and Schapire, 1995]. Note that because $\mathbb{E}[R] = o(T)$ the per round regret approaches zero. In the *bandit* version

of this problem the learning algorithm only observes $\ell^t(S^t)$ [Auer et al., 2003].

Our problem fits in this standard setting with $\mathcal{A}$ chosen to be the set of all ground set permutations $(v_1, v_2, \ldots v_n)$ and $\ell^t(S^t) \triangleq \ell(F^t, S^t)/n$. However, in this case $\mathcal{A}$ is very large so standard online learning algorithms which keep weight vectors of size $|\mathcal{A}|$ cannot be directly applied. Furthermore, our problem generalizes an NP-hard offline problem which has no polynomial time approximation scheme, so it is not likely that we will be able to derive any efficient algorithm with $o(T \ln |\mathcal{A}|)$ regret. We therefore instead consider α-regret, the loss incurred by the algorithm as compared to α times the best fixed prediction. $R_\alpha = \sum_{t=1}^T \ell^t(S^t) - \alpha \min_{S \in \mathcal{A}} \sum_{t=1}^T \ell^t(S)$. α-regret is a standard notion of regret for online versions of NP-hard problems. If we can show R_α grows sub linearly with T then we have shown loss converges to that of an offline approximation with ratio α.

Streeter and Golovin [2008] give online algorithms for the closely related problems of submodular function maximization and min-sum submodular set cover. In online submodular function maximization, the learning algorithm selects a set S^t with $|S^t| \leq k$ before F^t is revealed, and the goal is to maximize $\sum_t F^t(S^t)$. This problem differs from ours in that our problem is a loss minimization problem as opposed to an objective maximization problem. Online min-sum submodular set cover is similar to online submodular set cover except the loss is not cover time but rather

$$\hat{\ell}(F^t, S^t) \triangleq \sum_{i=0}^{n} \max(1 - F^t(S_i^t), 0). \tag{6.2}$$

Min-sum submodular set cover penalizes $1 - F^t(S_i^t)$ where submodular set cover uses $I(F^t(S_i^t) < 1)$. We claim that for certain applications the hard threshold makes more sense. For example, in repeated active learning problems minimizing $\sum_t \ell(F^t, S^t)$ naturally corresponds to minimizing the number of questions asked. Minimizing $\sum_t \hat{\ell}(F^t, S^t)$ does not have this interpretation as it charges less for questions asked when F^t is closer to 1. One might hope that minimizing ℓ could be reduced to or shown equivalent to minimizing $\hat{\ell}$. This is not likely to be the case, as the approximation algorithm of Streeter and Golovin [2008] does not carry over to online submodular set cover. Their online algorithm is based on approximating an offline algorithm which greedily maximizes $\sum_t \min(F^t(S), 1)$. Azar and Gamzu [2011] show that this offline algorithm, which they call the cumulative greedy

algorithm, does not achieve a good approximation ratio for average cover time.

Radlinski et al. [2008] consider a special case of online submodular function maximization applied to search result ranking. In their problem the objective function is assumed to be a binary valued submodular function with 1 indicating the user clicked on at least one document. The goal is then to maximize the number of queries which receive at least one click. For binary valued functions $\hat{\ell}$ and ℓ are the same, so in this setting minimizing the number of documents a user must view before clicking on a result is a min-sum submodular set cover problem. Our results generalize this problem to minimizing the number of documents a user must view before some possibly non-binary submodular objective is met. With non-binary objectives we can incorporate richer implicit feedback such as multiple clicks and time spent viewing results. Slivkins et al. [2010] generalize the results of Radlinski et al. [2008] to a metric space bandit setting.

Our work differs from the online set cover problem of Alon et al. [2003]; this problem is a single set cover problem in which the items that need to be covered are revealed one at a time. Kakade et al. [2007] analyze general online optimization problems with linear loss. If we assume that the functions F^t are all taken from a known finite set of functions $\mathcal{F}$ then we have linear loss over a $|\mathcal{F}|$ dimensional space. However, this approach gives poor dependence on $|\mathcal{F}|$.

6.3 Offline Analysis

We present an algorithm for online submodular set cover which extends the offline algorithm of Azar and Gamzu [2011] for the ranking with submodular valuations problem. Algorithm 1 shows this offline algorithm, called the adaptive residual updates algorithm. Here we use T to denote the number of objective functions and superscript t to index the set of objectives. This notation is chosen to make the connection to the proceeding online algorithm clear: our online algorithm will approximately implement Algorithm 1 in an online setting, and in this case the set of objectives in the offline algorithm will be the sequence of objectives in the online problem. The algorithm is a greedy algorithm similar to the standard algorithm for submodular set cover. The crucial difference is that instead of a normal gain term of

Algorithm 6.1 Offline Adaptive Residual

Input: Objectives $F^1, F^2, \ldots F^T$

Output: Sequence $S_1 \subset S_2 \subset \ldots S_n$

$S_0 \leftarrow \emptyset$

for $i \leftarrow 1 \ldots n$ **do**

$v \leftarrow \underset{v \in V}{\text{argmax}} \sum_t \hat{\delta}(F^t, S_{i-1}, v)$

$S_i \leftarrow S_{i-1} + v$

end for

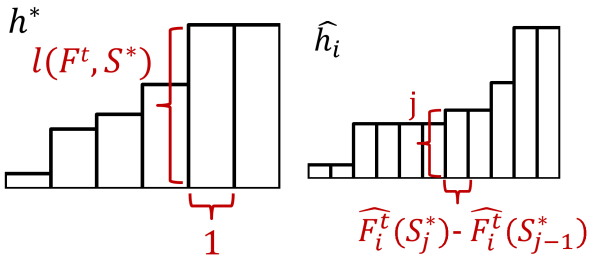

Figure 6.1: Histograms used in offline analysis

$F^t(S+v) - F^t(S)$ it uses a relative gain term

$$\hat{\delta}(F^t, S, v) \triangleq \begin{cases} \min(\frac{F^t(S+v) - F^t(S)}{1 - F^t(S)}, 1) & \text{if } F(S) < 1 \\ 0 & \text{otherwise} \end{cases}$$

The intuition is that (1) a small gain for F^t matters more if F^t is close to being covered ($F^t(S)$ close to 1) and (2) gains for F^t with $F^t(S) \geq 1$ do not matter as these functions are already covered. The main result of Azar and Gamzu [2011] is that Algorithm 1 is approximately optimal.

Theorem 6.1 (Azar and Gamzu [2011])**.** *The loss* $\sum_t \ell(F^t, S)$ *of the sequence produced by Algorithm 6.1 is within* $4(\ln(1/\delta) + 2)$ *of that of any other sequence.*

Here and throughout this chapter we use δ to denote the smallest non-zero gain of any of F^t. We note Azar and Gamzu [2011] allow for weights for each F^t. We omit weights for

simplicity. Also, Azar and Gamzu [2011] do not allow the sequence S to contain duplicates while we do–selecting a ground set element twice has no benefit but allowing them will be convenient for the online analysis. The proof of Theorem 6.1 involves representing solutions to the submodular ranking problem as histograms. Each histogram is defined such that the area of the histogram is equal to the loss of the corresponding solution. The approximate optimality of Algorithm 6.1 is shown by proving that the histogram for the solution it finds is approximately contained within the histogram for the optimal solution. In order to convert Algorithm 1 into an online algorithm, we will need a stronger version of Theorem 6.1. Specifically, we will need to show that when there is some additive error in the greedy selection rule Algorithm 6.1 is still approximately optimal.

For the optimal solution $S^* = \text{argmin}_{S \in V^n} \sum_t \ell(F^t, S)$ (V^n is the set of all size-n ordered sets of ground set elements), define a histogram h^* with T columns, one for each function F^t. Let the tth column have with width 1 and height equal to $\ell(F^t, S^*)$. Assume that the columns are ordered by increasing cover time so that the histogram is monotone non-decreasing. Note that the area of this histogram is exactly the loss of S^*.

For a sequence of sets $\emptyset = S_0 \subseteq S_1 \subseteq \dots S_n$ (e.g., those found by Algorithm 6.1) define a corresponding sequence of truncated objectives

$$\hat{F}_i^t(S) \triangleq \begin{cases} \min(\frac{F^t(S \cup S_{i-1}) - F^t(S_{i-1})}{1 - F^t(S_{i-1})}, 1) & \text{if } F^t(S_{i-1}) < 1 \\ 1 & \text{otherwise} \end{cases}$$

$\hat{F}_i^t(S)$ is essentially F^t except with (1) S_{i-1} given "for free", and (2) values rescaled to range between 0 and 1. We note that $\hat{F}_i^t$ is submodular and that if $F^t(S) \geq 1$ then $\hat{F}_i^t(S) \geq 1$. In this sense $\hat{F}_i^t$ is an easier objective than F^t. Also, for any v, $\hat{F}_i^t(\{v\}) - \hat{F}_i^t(\emptyset) = \hat{\delta}(F^t, S_{i-1}, v)$. In other words, the gain of $\hat{F}_i^t$ at $\emptyset$ is the normalized gain of F^t at S_{i-1}. This property will be crucial.

We next define truncated versions of h^*: $\hat{h}_1, \hat{h}_2, \dots \hat{h}_n$ which correspond to the loss of S^* for the easier covering problems involving $\hat{F}_i^t$. For each $j \in 1 \dots n$, let $\hat{h}_i$ have T columns of height j with the tth such column of width $\hat{F}_i^t(S_j^*) - \hat{F}_i^t(S_{j-1}^*)$ (some of these columns may have 0 width). Assume again the columns are ordered by height. Figure 6.1 shows h^* and $\hat{h}_i$.

We assume without loss of generality that $F^t(S_n^*) \geq 1$ for every t (clearly some choice of S^* contains no duplicates, so under our assumption that $F^t(V) \geq 1$ we also have $F^t(S_n^*) \geq 1$). Note that the total width of $\hat{h}_i$ is then the number of functions remaining to be covered after S_{i-1} is given for free (i.e., the number of F^t with $F^t(S_{i-1}) < 1$). It is not hard to see that the total area of $\hat{h}_i$ is $\sum_t \hat{\ell}(\hat{F}_i^t, S^*)$ where $\hat{l}$ is the loss function for *min-sum* submodular set cover (6.2).

$$\sum_t \hat{\ell}(\hat{F}_i^t, S^*) \leq \sum_t \ell(\hat{F}_i^t, S^*) \leq \sum_t \ell(F^t, S^*)$$

So we know $\hat{h}_i$ has area less than h^*. In fact, Azar and Gamzu [2011] show the following.

Lemma 6.1 (Azar and Gamzu [2011])**.** *$\hat{h}_i$ is completely contained within h^* when $\hat{h}_i$ and h^* are aligned along their lower right boundaries.*

We need one final lemma before proving the main result of this section. For a sequence S define $Q_i = \sum_t \hat{\delta}(F^t, S_{i-1}, v_i)$ to be the total normalized gain of the ith selected element and let $\Delta_i = \sum_{j=i}^n Q_j$ be the sum of the normalized gains from i to n. Define $\Pi_i = |\{t : F^t(S_{i-1}) < 1\}|$ to be the number of functions which are still uncovered before v_i is selected (i.e., the loss incurred at step i). Azar and Gamzu [2011] show the following result relating Δ_i to Π_i.

Lemma 6.2 (Azar and Gamzu [2011])**.** *For any i, $\Delta_i \leq (\ln 1/\delta + 2)\Pi_i$*

We now state and prove the main result of this section, that Algorithm 1 is approximately optimal even when the ith greedy selection is preformed with some additive error R_i. This theorem shows that in order to achieve low average cover time it suffices to *approximately* implement Algorithm 6.1. Aside from being useful for converting Algorithm 6.1 into an online algorithm, this theorem may be useful for applications in which the ground set V is very large. In these situations it may be possible to approximate Algorithm 6.1 (e.g., through sampling). Streeter and Golovin [2008] prove similar results for submodular function maximization and min-sum submodular set cover. Our result is similar, but the proof is non trivial. The loss function ℓ is highly non linear with respect to changes in $F^t(S_i^t)$, so it is conceivable that small additive errors in the greedy selection could have a

large effect. The analysis of Im and Nagarajan [2011] involves a version of Algorithm 6.1 which is robust to a sort of *multplicative* error in each stage of the greedy selection.

Theorem 6.2. *Let $S = (v_1, v_2, \ldots v_n)$ be any sequence for which*

$$\sum_t \hat{\delta}(F^t, S_{i-1}, v_i) + R_i \geq \max_{v \in V} \sum_t \hat{\delta}(F^t, S_{i-1}, v)$$

Then $\sum_t \ell(F^t, S^t) \leq 4(\ln 1/\delta + 2) \sum_t \ell(F^t, S^*) + n \sum_i R_i$

Proof. Let h be a histogram with a column for each Π_i with $\Pi_i \neq 0$. Let $\gamma = (\ln 1/\delta + 2)$. Let the ith column have width $(Q_i + R_i)/(2\gamma)$ and height $\max(\Pi_i - \sum_j R_j, 0)/(2(Q_i + R_i))$. Note that $\Pi_i \neq 0$ iff $Q_i + R_i \neq 0$ as if there are functions not yet covered then there is some set element with non zero gain (and vice versa). The area of h is

$$\sum_{i:\Pi_i \neq 0} \frac{1}{2\gamma}(Q_i + R_i) \frac{\max(\Pi_i - \sum_j R_j, 0)}{2(Q_i + R_i)} \geq \frac{1}{4\gamma} \sum_t \ell(F^t, S) - \frac{n}{4\gamma} \sum_j R_j$$

Here we used the fact that $\sum_{i:\Pi_i \neq 0} \Pi_i = \sum_t \ell(F^t, S)$. Assume h and every $\hat{h}_i$ are aligned along their lower right boundaries. We show that if the ith column of h has non-zero area then it is contained within $\hat{h}_i$. Then, it follows from Lemma 6.1 that h is contained within h^*, completing the proof.

Consider the ith column in h. Assume this column has non zero area so $\Pi_i \geq \sum_j R_j$. This column is at most $(\Delta_i + \sum_{j \geq i} R_j)/(2\gamma)$ away from the right hand boundary. To show that this column is in $\hat{h}_i$ it suffices to show that after selecting the first $k = \lfloor (\Pi_i - \sum_j R_j)/(2(Q_i + R_i)) \rfloor$ items in S^* we still have $\sum_t (1 - \hat{F}_i^t(S_k^*)) \geq (\Delta_i + \sum_{j \geq i} R_j)/(2\gamma)$. The most that $\sum_t \hat{F}_i^t$ can increase through the addition of one item is $Q_i + R_i$. Therefore, using the submodularity of $\hat{F}_i^t$,

$$\sum_t \hat{F}_i^t(S_k^*) - \sum_t \hat{F}_i^t(\emptyset) \leq k(Q_i + R_i) \leq \Pi_i/2 - \sum_j R_j/2$$

Therefore $\sum_t (1 - \hat{F}_i^t(S_k^*)) \geq \Pi_i/2 + \sum_j R_j/2$ since $\sum_t (1 - \hat{F}_i^t(\emptyset)) = \Pi_i$. Using Lemma 6.2

$$\Pi_i/2 + \sum_j R_j/2 \geq \Delta_i/(2\gamma) + \sum_j R_j/2 \geq (\Delta_i + \sum_{j \geq i} R_j)/(2\gamma)$$

□

Algorithm 6.2 Online Adaptive Residual

Input: Integer T

Initialize n online learning algorithms $E_1, E_2, \ldots E_n$ with $\mathcal{A} = V$

for $t = 1 \rightarrow T$ **do**

$\forall i \in 1 \ldots n$ predict v_i^t with E_i

$S^t \leftarrow (v_1^t, \ldots v_n^t)$

Receive F^t, pay loss $l(F^t, S^t)$

For E_i, $\ell^t(v) \leftarrow (1 - \hat{\delta}(F^t, S_{i-1}^t, v))$

end for

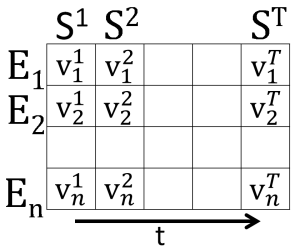

Figure 6.2: E_i selects the ith element in S^t.

6.4 Online Analysis

We now show how to convert Algorithm 6.1 into an online algorithm. We use the same idea used by Streeter and Golovin [2008] and Radlinski et al. [2008] for online submodular function maximization: we run n copies of some low regret online learning algorithm, $E_1, E_2, \ldots E_n$, each with action space $\mathcal{A} = V$. We use the ith copy E_i to select the ith item in each predicted sequence S^t. In other words, the predictions of E_i will be $v_i^1, v_i^2, \ldots v_i^T$. Figure 6.2 illustrates this. Our algorithm assigns loss values to each E_i so that, assuming E_i has low regret, E_i approximately implements the ith greedy selection in Algorithm 6.1. Algorithm 6.2 shows this approach. Note that under our assumption that $F^1, F^2, \ldots F^T$ is chosen by an oblivious adversary, the loss values for the ith copy of the online algorithm are oblivious to the predictions of that run of the algorithm. Therefore we can use any algorithm for learning against an oblivious adversary.

Theorem 6.3. *Assume we use as a subroutine an online prediction algorithm with expected regret* $\mathbb{E}[R] \leq \sqrt{T \ln n}$. *Algorithm 6.2 has expected* α*-regret* $\mathbb{E}[R_\alpha] \leq n^2\sqrt{T \ln n}$ *for* $\alpha = 4(\ln(1/\delta) + 2)$

Proof. Define a meta-action $\tilde{v_i}$ for the sequence of actions chosen by E_i, $\tilde{v_i} = (v_i^1, v_i^2, \ldots v_i^T)$. We can extend the domain of F^t to allow for meta-actions $F^t(S \cup \{\hat{v}_i\}) = F^t(S \cup \{v_i^t\})$. Let $\tilde{S}$ be the sequence of meta actions $\tilde{S} = (\tilde{v_1}, \tilde{v_2}, \ldots \tilde{v_n})$. Let R_i be the regret of E_i. Note that from the definition of regret and our choice of loss values we have that

$$\max_{v \in V} \sum_t \hat{\delta}(F^t, \tilde{S}_{i-1}, v) - \sum_t \hat{\delta}(F^t, \tilde{S}_{i-1}, \tilde{v}_i) = R_i$$

Therefore, $\tilde{S}$ approximates the greedy solution in the sense required by Theorem 6.2. Theorem 6.2 did not require that S be constructed V. From Theorem 6.2 we then have

$$\sum_t \ell(F^t, S^t) = \sum_t \ell(F^t, \tilde{S}) \leq \alpha \sum_t \ell(F^t, S^*) + n \sum_i R_i$$

The expected α-regret is then $\mathbb{E}[n \sum_i R_i] \leq n^2\sqrt{T \ln n}$ □

We describe several variations and extensions of this analysis, some of which mirror those for related work [Streeter and Golovin, 2008, Radlinski et al., 2008, Slivkins et al., 2010].

Avoiding Duplicate Items Since each run of the online prediction algorithm is independent, Algorithm 6.2 may select the same ground set element multiple times. This drawback is easy to fix. We can simply select any arbitrary $v_i \notin S_{i-1}$ if E_i selects a $v_i \in S_{i-i}$. This modification does not affect the regret guarantee as selecting a $v_i \in S_{i-1}$ will always result in a gain of zero (loss of 1). Therefore using a different item in place of a duplicate will always decrease regret.

Truncated Loss In some applications we only care about the first k items in the sequence S^t. For these applications it makes sense to consider a truncated version of $l(F^t, S^t)$ with parameter k

$$\ell^k(F^t, S^t) \triangleq \min\big(\{k\} \cup \{|S_i^t| : F^t(S_i^t) \geq 1\}\big)$$

This is cover time computed up to the kth element in S^t. The analysis for Theorem 6.2 also shows $\sum_t \ell^k(F^t, S^t) \leq 4(\ln 1/\delta + 2) \sum_t \ell(F^t, S^*) + k \sum_{i=1}^k R_i$. The proof is identical except

that we only use a k column histogram h, and the height of the columns is defined to be $\max(\Pi_i - \sum_{j=1}^{k} R_j, 0)/(2(Q_i + R_i))$ (the summation changed to only range over k). The area of this histogram is related to $\sum_t \ell^k(F^t, S^t)$ in the same way that the larger histogram is related to $\sum_t \ell(F^t, S^t)$. The corresponding regret bound is then $k^2\sqrt{T \ln n}$. Note here we are bounding truncated loss $\sum_t \ell^k(F^t, S^t)$ in terms of untruncated loss $\sum_t \ell(F^t, S^*)$. In this sense this bound is weaker. However, we replace n^2 with k^2 which may be much smaller. Algorithm 6.2 achieves this bound simultaneously for all k.

Multiple Objectives per Round Consider a variation of online submodular set cover in which instead of receiving a single objective F^t each round we receive a batch of objectives $F_1^t, F_2^t, \ldots F_m^t$ and incur loss $\sum_{i=1}^{m} \ell(F_i^t, S^t)$. In other words, each rounds corresponds to a ranking with submodular valuations problem. It is easy to extend Algorithm 6.2 to this setting by using $1 - (1/m)\sum_{i=1}^{m} \hat{\delta}(F_i^t, S_{i-1}^t, v)$ for the loss of action v in E_i. We then get $O(mk^2\sqrt{T \ln n})$ total regret–the regret of each of the online learning algorithms E_i is multiplied by m.

We note the originally published version of this work [Guillory and Bilmes, 2011a] mistakenly reported the regret for this setting as being $O(k^2\sqrt{mL^* \ln n} + m \ln n)$ where L^* is the total cover time of S^*. This was based on an incorrect application of the scaled loss bound of Cesa-Bianchi and Lugosi [2006] (Section 2.6). To correctly apply the scaled bound L^* should not be in terms of cover time but rather the losses of the individual online learning routines E_i. The correct bound presented here is simpler and actually tighter than this incorrect bound because $L^* \geq mT$.

Bandit Setting Consider a setting where instead of receiving full access to F^t we only observe the sequence of objective function values $F^t(S_1^t), F^t(S_2^t), \ldots F^t(S_n^t)$ (or in the case of multiple objectives per round, $F_i^t(S_j^t)$ for every i and j). We can extend Algorithm 6.2 to this setting by using a nonstochastic multiarmed bandits algorithm [Auer et al., 2003] for the E_i. With such an algorithm, E_i only requires access to $l^t(v_i^t)$ where v_i^t is the item selected by E_i at round t. In our case we only need to be able to compute $(1 - \hat{\delta}(F^t, S_{i-1}^t, v_i^t))$ and therefore it suffices to observe $F^t(S_1^t), F^t(S_2^t), \ldots F^t(S_n^t)$. The typical regret bound for these algorithms adds a factor of $O(\sqrt{n})$ to the full information bound. We therefore have total regret $O(k^2\sqrt{nT \ln n})$

We note duplicate removal becomes more subtle in the bandit setting: should we feedback a gain of zero when a duplicate is selected or the gain of the non-duplicate replacement? We propose either is valid if replacements are chosen obliviously. By oblivious we mean the way in a replacement for the tth selection of E_i is chosen does not depend on the previous selections of E_i. To see this note that if we choose replacements obliviously then it as though we have obliviously modified the loss vectors for E_i. Assuming that E_i has low regret against any oblivious adversary, E_i still guarantees low regret after these modifications to the loss vector.

Bandit Setting with Expert Advice We can further generalize the bandit setting to the contextual bandit setting [Langford and Zhang, 2007] (e.g., the bandit setting with expert advice [Auer et al., 2003]). Say that we have access at time step t to predictions from a set of m experts. Let $\tilde{v}_j$ be the meta action corresponding to the sequence of predictions from the jth expert and $\tilde{V}$ be the set of all $\tilde{v}_j$. Assume that E_i guarantees low regret with respect to $\tilde{V}$

$$\sum_t \hat{\delta}(F^t, S^t_{i-1}, v^t_i) + R_i \geq \max_{\tilde{v} \in \tilde{V}} \sum_t \hat{\delta}(F^t, S^t_{i-1}, \tilde{v}) \tag{6.3}$$

where we have extended the domain of each F^t to include meta actions as in the proof of Theorem 6.3. Additionally assume that $F^t(\tilde{V}) \geq 1$ for every t. In this case we can show $\sum_t \ell^k(F^t, S^t) \leq \min_{S^* \in \tilde{V}^m} \sum_t \ell^m(F^t, S^*) + k \sum_{i=1}^k R_i$. The Exp4 algorithm [Auer et al., 2003] has $R_i = O(\sqrt{nT \ln m})$ giving total regret $O(k^2\sqrt{nT \ln m})$. The only change to the analysis is that we use $\tilde{V}$ in place of the original ground set V. Experts may use context in forming recommendations. For example, in a search ranking problem the context could be the query.

More General Cost Functions Azar and Gamzu [2011] discuss a technique for reducing min-sum submodular set cover to ranking with submodular valuations. The idea behind this reduction is to make multiple thresholded copies of F ($\min(F(S), \alpha_1)$, $\min(F(S), \alpha_2)$, etc.) If we were to create infinitely many such copies with thresholds chosen uniformly from $[0, 1]$, then in the limit the average cover time of this collection of objectives would equal the min-sum submodular set cover cost function (6.2). By using finitely many copies we can approximate the min-sum set cover cost within aribtrary percision. This idea can then

be generalized to other cost functions In particular, Azar and Gamzu [2011] claim their analysis can be applied to any cost of the form $\ell(F, S) = \sum_i c(\min(F(S_i), 1))$ where c is a non-increasing, non-negative cost function. Such a loss could be used to, for example, interpolate between the smooth min-sum loss and the hard threshold cover time loss.

This same technique could potentially be applied to the online setting using the variation of the online problem with multiple objectives per round. There are a few potential obstacles to this extension. First, it may be necessary to extend our analysis to allow for non-uniform weights on the objectives (recall we omitted this aspect of the offline problem for simplicity). Second, our regret bound grows with the number of objectives per round, so care must be taken to balance this growth with the quality of the approximation.

6.5 Experiments

We present experimental results which validate our theory both on a synthetic example and a repeated active learning movie recommendation problem.

6.5.1 Synthetic Example

We first present a synthetic example for which the online cumulative greedy algorithm [Streeter and Golovin, 2008] fails, based on the example in Azar and Gamzu [2011] for the offline setting. Consider an online ad placement problem where the ground set V is a set of available ad placement actions (e.g., a $v \in V$ could correspond to placing an ad on a particular web page for a particular length of time). On round t, we receive an ad from an advertiser, and our goal is to acquire λ clicks for the ad using as few advertising actions as possible. Define $F^t(S_i^t)$ to be $\min(c_i^t, \lambda)/\lambda$ where c_i^t is number of clicks acquired from the ad placement actions S_i^t.

Say that we have n advertising actions of two types: 2 broad actions and $n-2$ narrow actions. Say that the ads we receive are also of two types. Common type ads occur with probability $(n-1)/n$ and receive 1 and $\lambda - 1$ clicks respectively from the two broad actions and 0 clicks from narrow actions. Uncommon type ads occur with probability $1/n$ and receive λ clicks from one randomly chosen narrow action and 0 clicks from all other actions. Assume $\lambda \geq n^2$. Intuitively broad actions could correspond to ad placements on sites for

which many ads are relevant. The optimal strategy giving an average cover time $O(1)$ is to first select the two broad actions covering all common ads then select the narrow actions in any order. However, the offline cumulative greedy algorithm will pick all narrow actions before picking the broad action with gain 1 giving average cover time $O(n)$.

The left of Figure 6.3 shows average cover time for our proposed algorithm and the cumulative greedy algorithm of [Streeter and Golovin, 2008] on the same sequences of random objectives. For this example we use $n = 25$ and the bandit version of the problem with the Exp3 algorithm [Auer et al., 2003]. We also plot the average cover times for offline solutions as baselines. As seen in the figure, the cumulative algorithms converge to higher average cover times than the adaptive residual algorithms. Interestingly, the online cumulative algorithm does better than the offline cumulative algorithm: it seems added randomization helps.

6.5.2 Repeated Active Learning for Movie Recommendation

Consider a movie recommendation website such as in Section 4.6 which asks users a sequence of questions before they are given recommendations. We define an online submodular set cover problem for repeatedly choosing sequences of questions in order to quickly eliminate a large number of movies from consideration. This is similar conceptually to the diagnosis problem discussed in the introduction. Define the ground set V to be a set of questions (for example "Do you want to watch something released in the past 10 years?" or "Do you want to watch something from the Drama genre?"). Define $F^t(S)$ to be proportional to the number of movies eliminated from consideration after asking the tth user S. Specifically, let H be the set of all movies in our database and $V^t(S)$ be the subset of movies consistent with the tth user's responses to S. Define $F^t(S) \triangleq \min(|H \setminus V^t(S)|/c, 1)$ where c is a constant. $F^t(S) \geq$ iff after asking the set of questions S we have eliminated at least c movies.

We set H to be a set of 11634 movies available on Netflix's Watch Instantly service and use 803 questions based on those we used in Section 4.6 for single round learning and covering. To simulate user responses to questions, on round t we randomly select a movie from H and assume the tth user answers questions consistently with this movie. We set

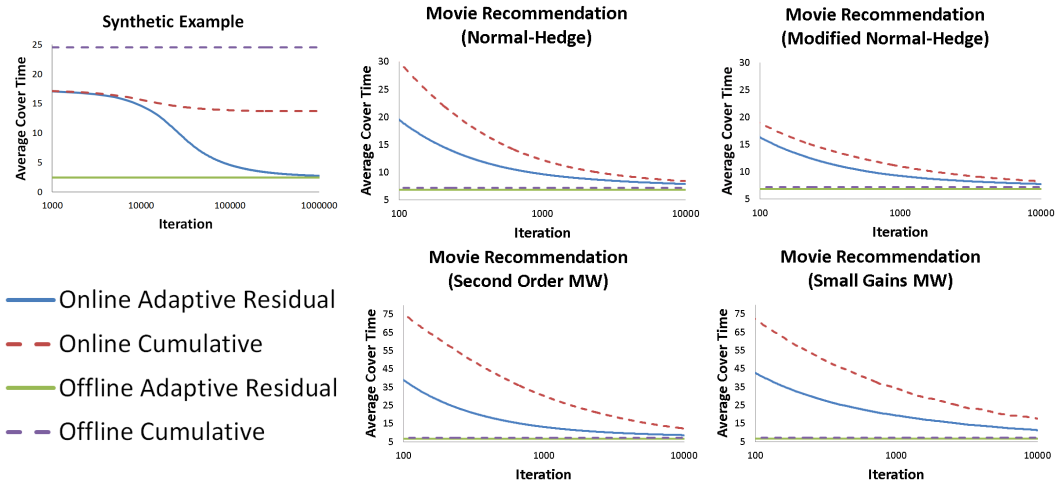

Figure 6.3: Average cover time

$c = |H| - 500$ so the goal is to eliminate about 95% of all movies. We evaluate in the full information setting: this makes sense if we assume we receive as feedback the movie the user actually selected. As our online prediction subroutine we tried Normal-Hedge [Chaudhuri et al., 2009], a second order multiplicative weights method [Cesa-Bianchi et al., 2007], and a version of multiplicative weights for small gains using the doubling trick (Section 2.6 of [Cesa-Bianchi and Lugosi, 2006]). We also tried a heuristic modification of Normal-Hedge which fixes $c_t = 1$ for a fixed, more aggressive learning rate than theoretically justified.

The right of Figure 6.3 shows average cover time for 100 runs of $T = 10000$ iterations. Note the different scale in the bottom row–these methods performed significantly worse than Normal-Hedge. The online cumulative greedy algorithm converges to a average cover time close to but slightly worse than that of the adaptive greedy method. The differences are more dramatic for prediction subroutines that converge slowly. The modified Normal-Hedge has no theoretical justification, so it may not generalize to other problems. For the modified Normal-Hedge the final average cover times are 7.72 adaptive and 8.22 cumulative. The offline values are 6.78 and 7.15.

Chapter 7

CONCLUSION

We have presented a number of results connecting active learning and submodular function optimization. An additional common theme in these results is a focus on non traditional learning settings. All of our results consider *non stochastic* settings where no assumptions are made about the way in which the unlabeled data is generated. This is in contrast to the standard statistical learning setting where unlabeled data points are assumed to be independent and identically distributed. Outside of online learning, there is relatively little work on non stochastic learning settings. The batch setting considered in Chapter 5 and the repeated active earning setting in Chapter 6 are also relatively unexplored settings. This dissertation can therefore be taken as specific evidence of a general idea: submodular optimization is a useful tool for expanding our understanding of machine learning.

We conclude by mentioning a few of the many interesting remaining problems.

Both Chapter 4 and 5 use submodular functions to implicitly define a hypothesis class. However, the way in which this implicit hypothesis class is defined is different in each chapter. In the interactive setting (Chapter 4) an explicit hypothesis class is expanded by monotone submodular objectives $G_h(S)$ and a threshold κ where (G_h, κ) is assumed adversarially uncoverable. In the batch setting (Chapter 5) a symmetric submodular function is used as a regularizer. Can these two different kinds of assumptions be unified? Can symmetric submodular functions be used as regularizers in the interactive setting?

Related, the use of explicit noise thresholds in Chapter 4 is not always practical. Recall that the problem in this chapter is to achieve coverage $F_h(S) = F_h(Q \times R)$ for every $h \in H$ with $G_h(S^*) < \kappa$. G_h and κ define which hypotheses are close to h^*. An alternative problem which avoids the explicit threshold κ is to achieve coverage $F_{\hat{h}}(S) = F_{\hat{h}}(Q \times R)$ for only the hypothesis $\hat{h}$ which is closest to h^* (for some definition of closeness). Are there approximation algorithms for this problem? It may be possible to use the same algorithm

in Chapter 4 along with a strategy for finding the right threshold κ. However, a difficulty with this approach is that our algorithm only guarantees $F_h(S) = F_h(Q \times R)$ if $G_h(S^*) < \kappa$. It does not necessarily verify whether or not $G_h(S^*) < \kappa$ for any h. A strategy for finding the right value of κ would need to verify this. There are similar open questions for the approximate active learning problem in Chapter 3 which uses an explicit threshold.

The repeated active learning algorithm in Chapter 6 picks a fixed sequence of questions to ask at the start of each round of active learning. Question responses influence the questions asked on proceeding rounds, but within each round the algorithm is in some sense batch: within a round, responses to questions do not influence the choice of questions. Can we perform repeated active learning with question asking strategies which are interactive within each round?

BIBLIOGRAPHY

P. Afshani, E. Chiniforooshan, R. Dorrigiv, A. Farzan, M. Mirzazadeh, N. Simjour, and H. Zarrabi-Zadeh. On the complexity of finding an unknown cut via vertex queries. In *COCOON*, 2007.

P. Alimonti and V. Kann. Hardness of approximating problems on cubic graphs. *Algorithms and Complexity*, pages 288–298, 1997.

N. Alon, B. Awerbuch, and Y. Azar. The online set cover problem. In *STOC*, 2003.

A. Asadpour, H. Nazerzadeh, and A. Saberi. Stochastic submodular maximization. In *Workshop on Internet and Network Economics*, 2008.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2003.

Y. Azar and I. Gamzu. Ranking with Submodular Valuations. In *SODA*, 2011.

F. Bach. Structured sparsity-inducing norms through submodular functions. *NIPS*, 2010.

M. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *ICML*, 2006.

M. Balcan, A. Z. Broder, and T. Zhang. Margin based active learning. In *COLT*, 2007.

M. Balcan, S. Hanneke, and J. Vaughan. The true sample complexity of active learning. *Machine learning*, 80(2), 2010.

J. Balcázar, J. Castro, D. Guijarro, J. Köbler, and W. Lindner. A general dimension for query learning. *Journal of Computer and System Sciences*, 73(6):924–940, 2007.

G. Bellala, S. Bhavnani, and C. Scott. Extensions of generalized binary search to group identification and exponential costs. In *NIPS*, 2010.

Y. Bengio, O. Delalleau, and N. Le Roux. Label propagation and quadratic criterion. In O. Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-Supervised Learning*. MIT Press, 2006.

A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *ICML*, 2009.

A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *ICML*, 2001.

A. Blum, J. Lafferty, M. R. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. In *ICML*, 2004.

J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, 1998.

G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular set function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6), 2011.

N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Worst-case analysis of selective sampling for linear classification. *JMLR*, 7, 2006.

N. Cesa-Bianchi, Y. Mansour, and G. Stoltz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 2007.

N. Cesa-Bianchi, C. Gentile, and F. Vitale. Fast and optimal prediction of a labeled tree. In *COLT*, 2009.

N. Cesa-Bianchi, C. Gentile, F. Vitale, and G. Zappella. Active Learning on Graphs via Spanning Trees. In *NIPS Workshop on Networks Across Disciplines*, 2010a.

N. Cesa-Bianchi, C. Gentile, F. Vitale, and G. Zappella. Active learning on trees and graphs. In *COLT*, 2010b.

O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised learning*. MIT press, 2006.

K. Chaudhuri, Y. Freund, and D. Hsu. A parameter-free hedging algorithm. In *NIPS*, 2009.

W. Cunningham. Optimal attack and reinforcement of a network. *Journal of the ACM*, 1985.

S. Dasgupta. Analysis of a greedy active learning strategy. In *NIPS*, 2004.

S. Dasgupta. Coarse sample complexity bounds for active learning. In *NIPS*, 2006.

S. Dasgupta, W. Lee, and P. Long. A theoretical analysis of query selection for collaborative filtering. *Machine Learning*, 51(3), 2003.

S. Dasgupta, A. T. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. In *COLT*, 2005.

S. Dasgupta, D. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. In *NIPS*, 2007.

B. Dean, M. Goemans, and J. Vondrak. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *FOCS*, 2004.

E. Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Mathematics Doklady*, 11(5):1277–1280, 1970.

J. Edmonds. Submodular functions, matroids, and certain polyhedra. *Combinatorial structures and their applications*, pages 69–87, 1970.

J. Edmonds and R. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4), 1998.

U. Feige, V. S. Mirrokni, and J. Vondrák. Maximizing non-monotone submodular functions. In *FOCS*, 2007.

M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functionsii. In *Polyhedral Combinatorics*, volume 8 of *Mathematical Programming Studies*, pages 73–87. Springer, 1978.

Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37, 1995.

S. Fujishige. *Submodular Functions and Optimization.* 2005.

S. Fujishige, T. Hayashi, and S. Isotani. The minimum-norm-point algorithm applied to submodular function minimization and linear programming, 2006.

M. Garey and D. Johnson. *Computers and intractability. A guide to the theory of NP-completeness.* 1979.

M. Goemans and J. Vondrák. Stochastic covering and adaptivity. In *LATIN*, 2006.

D. Golovin and A. Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization. In *COLT*, 2010.

D. Golovin, M. Faulkner, and A. Krause. Online distributed sensor selection. In *IPSN*, 2010a.

D. Golovin, A. Krause, and D. Ray. Near-optimal bayesian active learning with noisy observations. In *NIPS*, 2010b.

M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

A. Guillory and J. Bilmes. Label Selection on Graphs. In *NIPS*, 2009.

A. Guillory and J. Bilmes. Interactive submodular set cover. In *ICML*, 2010a.

A. Guillory and J. Bilmes. Interactive submodular set cover, 2010b. Technical Report. UWEETR-2010-0001.

A. Guillory and J. Bilmes. Simultaneous learning and covering with adversarial noise. In *NIPS 2010 Workshop on Discrete Optimization in Machine Learning (DISCML)*, 2010c.

A. Guillory and J. Bilmes. Online Submodular Set Cover, Ranking, and Repeated Active Learning. In *NIPS*, 2011a.

A. Guillory and J. Bilmes. Simultaneous learning and covering with adversarial noise. In *ICML*, 2011b.

A. Guillory and J. Bilmes. Active Semi-Supervised Learning using Submodular Functions. In *UAI*, 2011c.

A. Gupta, V. Nagarajan, and R. Ravi. Thresholded Covering Algorithms for Robust and Max-Min Optimization. In *ICALP*, 2010a.

A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. *WINE*, 2010b.

S. Hanneke. The cost complexity of interactive learning, 2006. Unpublished.

S. Hanneke. A bound on the label complexity of agnostic active learning. In *ICML*, 2007.

E. Hazan and S. Kale. Online submodular minimization. In *NIPS*, 2009.

M. Herbster and G. Lever. Predicting the Labelling of a Graph via Minimum p-Seminorm Interpolation. In *COLT*, 2009.

S. Hoi, R. Jin, J. Zhu, and M. Lyu. Batch mode active learning and its application to medical image classification. In *ICML*, 2006.

S. Im and V. Nagarajan. Minimum Latency Submodular Cover. *ArXiv e-prints*, 2011.

S. Iwata and J. Orlin. A simple combinatorial algorithm for submodular function minimization. In *SODA*, 2009.

S. Jegelka and J. Bilmes. Online submodular minimization for combinatorial structures. In *ICML*, 2011.

A. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class batch-mode active learning for image classification. In *ICRA*, 2010.

S. M. Kakade, A. T. Kalai, and K. Ligett. Playing games with approximation algorithms. In *STOC*, 2007.

G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 1999.

D. Kempe, J. Kleinberg, and E. Tardos. Influential nodes in a diffusion model for social networks. In *ICALP*, 2005.

P. Kohli, L. Ladickỳ, and P. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

A. Krause, H. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *JMLR*, 2008a.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, 2008b.

J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *NIPS*, 2007.

J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko. Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM Journal on Discrete Mathematics*, 23(4), 2010.

H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *NAACL/HLT*, 2010.

H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *HLT*, 2011.

L. Lovász. Submodular functions and convexity. *Mathematical programming: the state of the art*, 1983.

M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, pages 234–243, 1978.

M. Narasimhan, N. Jojic, and J. Bilmes. Q-clustering. In *NIPS*, 2006.

H. Narayanan. *Submodular Functions and Electrical Networks*. 1997.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14:265–294, 1978.

F. Orabona and N. Cesa-Bianchi. Better algorithms for selective sampling. In *ICML*, 2011.

M. Queyranne. Minimizing symmetric submodular functions. *Mathematical Programming*, 82:3–12, 1998.

F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *ICML*, 2008.

H. Schütze, E. Velipasaoglu, and J. O. Pedersen. Performance thresholding in practical text classification. In *CIKM*, 2006.

P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

L. Shi, Y. Zhao, and J. Tang. Batch mode active learning for networked data. *ACM Transactions on Intelligent Systems and Technology*, 2010.

A. Slivkins, F. Radlinski, and S. Gollapudi. Learning optimally diverse rankings over large document collections. In *ICML*, 2010.

P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *NIPS*, 2010.

M. Streeter and D. Golovin. An online algorithm for maximizing submodular functions. In *NIPS*, 2008.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *JMLR*, 2, 2001.

V. Vazirani. *Approximation algorithms*. Springer Verlag, 2001.

L. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4), 1982.

H. Yang and D. Wong. Efficient Network Flow Based Min-Cut Balanced Partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15 (12):1533, 1996.

VITA

Andrew Guillory earned a Bachelor of Science in Computer Science from the Georgia Institute of Technology in 2006. At the University of Washington he earned a Master of Science in Computer Science and Engineering in 2008 and Doctor of Philosophy in Computer Science and Engineering in 2012.