Spring 5-20-2017

An Improved Algorithm for Learning to Perform Exception-Tolerant Abduction

Mengxue Zhang
Washington University in St Louis

Washington University in St. Louis

School of Engineering and Applied Science

Department of Computer Science and Engineering

Thesis Examination Committee:
Brendan Juba
Roman Garnett
Benjamin Moseley

An Improved Algorithm for Learning to Perform Exception-Tolerant Abduction

by

Mengxue Zhang

A thesis presented to the School of Engineering and Applied Science
of Washington University in partial fulfillment of the
requirements for the degree of

Master of Science

May 2017
Saint Louis, Missouri

Contents

List of Tables

List of Figures

Acknowledgments

B. Juba is supported by an AFOSR Young Investigator Award. I would like to thank Prof. Juba for giving me the chance to do the research and advising me for the Master Thesis.

Mengxue Zhang

Washington University in Saint Louis
May 2017

ABSTRACT OF THE THESIS

An Improved Algorithm for Learning to Perform Exception-Tolerant Abduction

by

Mengxue Zhang

Master of Science in Computer Science

Washington University in St. Louis, May 2017

Research Advisor: Professor Katherine Doe

Inference from an observed or hypothesized condition to a plausible cause or explanation for this condition is known as *abduction.* For many tasks, the acquisition of the necessary knowledge by machine learning has been widely found to be highly effective. However, the semantics of learned knowledge are weaker than the usual classical semantics, and this necessitates new formulations of many tasks. We focus on a recently introduced formulation of the abductive inference task that is thus adapted to the semantics of machine learning. A key problem is that we cannot expect that our causes or explanations will be perfect, and they must tolerate some error due to the world being more complicated than our formalization allows. This is a version of the *qualification problem*, and in machine learning, this is known as *agnostic* learning. In the work by Juba that introduced the task of learning to make abductive inferences, an algorithm is given for producing k-DNF explanations that tolerates such exceptions: if the best possible k-DNF explanation fails to justify the condition with probability ϵ, then the algorithm is promised to find a k-DNF explanation that fails to justify the condition with probability at most $O(n^k\epsilon)$, where n is the number of propositional attributes used to describe the domain. Here, we present an improved algorithm for this task. When the best k-DNF fails with probability ϵ, our algorithm finds a k-DNF that fails with probability at most $\tilde{O}(\sqrt{n^k}\epsilon)$ (i.e., suppressing logarithmic factors in n and $1/\epsilon$).We examine the empirical advantage of this new algorithm over the previous algorithm in two test domains, one of explaining conditions generated by a "noisy" k-DNF rule, and another of explaining conditions that are actually generated by a linear threshold rule.

We also apply the algorithm on the real world application Anomaly explanation.In this work,as opposed to anomaly detection, we are interested in finding possible descriptions of what may be causing anomalies in visual data. We use PCA to perform anomaly detection. The task is attaching semantics drawn from the image meta-data to a portion of the anomalous images from some source such as web-came. Such a partial description of the anomalous images in terms of the meta-data is useful both because it may help to explain what causes the identified anomalies, and also because it may help to identify the truly unusual images that defy such simple categorization. We find that it is a good match to apply our approximation algorithm on this task. Our algorithm successfully finds plausible explanations of the anomalies. It yields low error rate when the data set is large(>80,000 examples) and also works well when the data set is not very large(< 50,000 examples). It finds small 2-DNFs that are easy to interpret and capture a non-negligible.

Chapter 1

Introduction

1.1 Abduction Reasoning

Reasoning is the process of using existing knowledge to draw conclusions, make predictions, or construct explanations. There are three types of reasoning common in use: deductive reasoning, inductive reasoning and abductive reasoning. Deductive reasoning is the process of reasoning from some statements to reach a logically certain conclusion. For example, Wendy always sit on chair, if there is no chair in the room, we can deduce that Wendy is standing in the room. Inductive reasoning is a logical process in which there are multiple premises, all believed true or were been true at high probability, are combined to obtain a specific conclusion.

Abductive reasoning, one of the most important terms in this thesis, is the process of inferring a reasonable explanation for an observation or hypothetical situation.For example, suppose a man walks into a hotel and his clothing is wet. We may naturally assume that it is raining outside. This might not be true, and his clothing may have gotten wet some other way, but it is the most reasonable explanation of the given facts. In other words, abductive reasoning tends to represent at least a highly plausible explanation for the given facts. Abduction is powerful—in enabling us to find hidden explanation of events, it furthermore enables us to generate new theories. Abductive reasoning can be applied in diverse problems, such as image understanding [10, 31], natural language understanding [14], plan recognition [9], and so on.

Although most early work on abduction relied on explicit knowledge engineering to capture the domains in which such inference was to be performed, much knowledge engineering has been replaced by machine learning. The reasons can be explained roughly as follows: the main lessons of the CYC project [21] were that *(i)* the scope of knowledge needed to support ordinary human inferences is vast, and would take many decades to formalize in its entirety and *(ii)* such a large knowledge engineering effort seems to inevitably suffer from semantic drift and consequently, brittleness. Machine learning is a means to circumvent both of these problems. The price of using learned knowledge is that its semantics are inevitably weaker than those of classical knowledge. So, while these weaker semantics may grant us some additional robustness (as has been argued, for example, by Valiant [35, 36]), they also require us to reconsider the foundations of the various tasks we wish to perform. A surprising benefit of this exercise is that it turns out that a *combined* learning and reasoning task may be easier than either of its constituent parts: Khardon and Roth [18] demonstrated that algorithms for such combined tasks may efficiently learn and reason with representations that would be intractable to learn or reason about using standalone algorithms. Motivated by these advantages, we will likewise consider a combined learning and abductive reasoning task.

1.2 Abduction Task

In particular, we focus on a new formulation of abductive reasoning introduced by Juba [17] based on PAC-learning [34]. In this model, learning is accomplished using *examples* that consist of settings of each of the various Boolean attributes. For example, if our domain is reasoning about people, then our attributes may include “female” (yes or no), “male” (yes or no), “brown hair” (yes or no), “taller than 1.5m” (yes or no) and so on. Each example corresponds to a person, and consists of a setting of all of these attributes. All of the examples contain the same attributes, but they may be set differently. In the model, these examples are drawn from an arbitrary distribution D over $\{0, 1\}^n$ (for our n attributes). The abduction task is then, given a *condition* that we wish to explain, that is captured by a Boolean formula c, to use these examples to find a *hypothesis* formula h that explains the condition in the following sense. h should (approximately) entail c, that is, when h is true, c should almost always also be true; and, h itself should be true as often as possible, i.e., we wish to find the most likely such h. For example, the query might indicate whether or not the stated facts of the story hold in a specific example, while h is some other (most likely) condition that yields the given condition.

A key problem with such formalizations is that we cannot expect our explanations to be perfect. Those explanations must tolerate some errors – situations where the explanation should hold, but the condition in question fails to materialize – due to our failure to model the real world in every last detail. This is essentially a variant of the *qualification problem* [23]. In the work by Juba that introduced this task, an algorithm is given for producing k-DNF explanations that features some tolerance to such exceptions: if, under the best possible k-DNF explanation, the condition fails to materialize with probability ϵ, then the algorithm is promised to find a k-DNF explanation under which the condition only fails to materialize with probability at most $O(n^k\epsilon)$ (where again, n is the number of propositional attributes). Of course, this could be quite a bit larger than the best probability ϵ, and we would like to reduce (if not eliminate) this dependence on n.

1.3 Brief Summary for the Theory of Algorithm

In this work, we introduce an improved algorithm for this task which finds a k-DNF explanation that fails with probability at most $\tilde{O}(\sqrt{n^k}\epsilon)$[1]. Our algorithm is an extension of an earlier algorithm by Peleg [29] for the closely related *"Red-Blue Set Cover" problem* [7]. Roughly, in such a problem, we are given a collection of sets that we wish to use to cover all of the "blue" elements while covering as few "red" elements as possible. The correspondence then, is that we assign every example a blue element, while assigning examples in which the desired condition fails to hold a "red" element, and take the possible terms of size k (for a k-DNF) as our collection of sets. Now, our task is to choose terms of size k that "cover" *as many* blue elements as possible while covering as few red elements as possible—the main difference is that we no longer require covering *all* of the blue elements. More precisely, in the variant that is relevant to us, we are given a target fraction μ of the blue elements to cover (less than 1), and we seek to minimize the *ratio of red-to-blue elements we cover* in meeting this objective. Thus our task is actually also distinct from the *"positive-negative*

[1]That is, ignoring logarithmic factors.

partial set cover" problem studied by Miettinen [25], in which one wishes to minimize the *sum* of the number of negative (red) elements covered and the number of positive (blue) elements uncovered.

At the heart of Peleg's algorithm is an approximation algorithm for the weighted set cover problem; in our extension, this standard weighted set cover problem is instead a *partial* set cover problem. Slavík [33] had already shown that the greedy algorthm achieves the same approximation ratio for such a variant of weighted set cover, so we are able to easily complete the rest of the analysis after this modification. The resulting algorithm increases the error by a $O(\sqrt{n^k \log \frac{n+\log 1/\delta}{\epsilon}})$ factor, where again ϵ is the error rate achieved by the best explanation that is true with probability at least the target μ, and δ is the probability that we fail on account of drawing an unrepresentative set of examples.

We also investigate the empirical advantage of this new algorithm over the previous, "Tolerant Elimination" algorithm considered by Juba. We consider two test domains. In the first domain, there is a "planted" k-DNF rule that is used to define the condition, subject to some independent random noise. Thus, in this case, we have a good sense of what the ideal error rate should be. We find that both algorithms perform well at this simple task. In the second domain, the condition is actually defined by a (random) linear threshold rule. We know that in general, such linear threshold rules cannot be approximated well by a k-DNF, and so this domain exercises the algorithms' ability to tolerate errors that are due to the actual condition being too complex for our formalism to capture. We find that Tolerant Elimination completely fails at this task, never achieving an error rate lower than the trivial rule that is always satisfied, whereas our new algorithm is able to identify rules that are satisfied with controllable probabilities, that achieve substantially lower error rates.

1.4 Apply the Algorithm for Solving Anomaly Explanation Task

1.4.1 Problem Introduction

An anomaly is an observation that does not conform to expected behavior. The usual version of anomaly detection involves an algorithm to identify these anomalies, which are then subject to further investigation, for example by a human analyst. This work offers an option that, understanding what might have caused these anomalies. Once anomalies have been detected, we propose a way to find structure within those anomalies to better understand them. To reiterate, an anomaly is based on creating a model of expected variations in a scene and then defining examples that don't fit this model as anomalies.

For example, suppose you have a collection of traffic camera images, and you want to better understand what comprises atypical traffic conditions. You could obtain a collection of metadata corresponding to these images. Similar as I mentioned before, there are some attributes might include weather patterns, rush hour periods, holidays and so on. We propose to consider the task of *identifying what metadata conditions are associated with anomalous data.* For instance, the co-occurrence of rush hour and snowy weather be correlated in anomalous images, as these conditions could result in traffic delays, low visibility, or accidents. There may be other anomalies that you might be able to identify, but unable to easily explain from a metadata collection, such as a power outage, or snow covering the camera lens. These, you would perhaps take a separate look at to see if they are relevant. This is the type of application we envision.

In sum, given a list of metadata attributes about dataset, differences in the metadata can be used to infer an informative relationship. Our anomaly explanation seeks to attach semantics drawn from the image metadata to a portion of the anomalous images by returning a list of conditions on metadata that are associated with anomalies. Not all metadata attributes necessarily have conditions, as not all are necessarily associated with anomalies. In this work, we emphasized approximate validity in our descriptions of anomalies. We want the conditions we obtain to reliably indicate that data satisfying those conditions will be anomalous. In this sense, our description of an anomalous event would be informative: when these conditions hold, the image should be an anomaly. Thus, we aim to find a set of anomalies that are easily explainable. These explainable anomalies may the focus of study, or they may be more common anomalies that keep occurring that a user wants to filter out in order to focus on more interesting cases.

Our Partial Red-Blue cover greedy algorithm is a good fit to solving this task. With training set containing more than 80k images: given such large training set, it was able to find simple conditions that explain between 1/10 and 1/6 of the anomalies with precision greater than 75% and 1/30 of the anomalies with precision greater than 97%. With training sets of size closer to 50k, the algorithm works well but were not able to obtain greater than 60% precision, and thus could only address this task weakly at best. We note that since our task is inherently one of learning about events that occur by definition at most 3% of the time, it is not surprising that they should require a large traning set(and indeed, also large test set).

1.4.2 Data-set

The data combines webcam image data with metadata collected from a variety of sources. These metadata concern the semantic contents of the image (i.e., image labels obtained via object recognition) and local conditions when the image was collected.

We use webcam data from four locations in the AMOS database [15], which takes a photo from each webcam approximately every thirty minutes. We selected these locations based on three criteria: first, they are very stable; second, they have a relatively large number of images for the AMOS collection; and third, they are located in the USA, and so data on the weather and holidays at these locations was readily available from common sources. For each camera, there were between 73847 and 131873 images, and the metadata ranged from 195 to 335 dimensions. The locations are a pond (camera #269), Lake Mono (#4312), the Moody Gardens theme park (#623), and a Toledo highway (#21656). The actual longitude and latitude of the cameras have been estimated using prior work by Jacobs et al. [16].

To gather metadata, we use the Google Vision API [3], the SunCalc API [2], the Python holidays library [32], and data from The Weather Channel [8] to generate a binary and nonbinary metadata collection. The binary version is intended to provide a “summary” of nonbinary variables for those methods that require all variables to be binary.

We use the label detection feature in the Google Vision API to obtain a list of objects found in images. Each label was a variable. If an image did not receive that label, then the variable takes the value 0. If it received the label, the variable takes the value 1 in the binary dataset, and the value of the score in the nonbinary dataset. The score is a value between 0 and 1 that represents the confidence that the label is relevant to the image. Google Vision failed to run on a small number of the images. For these, we mark all Google Vision variables as 0 and for all images, we append an additional variable to signify whether or not Google Vision successfully ran.

We use the SunCalc API to create a list of binary variables for each image based on whether or not it was taken within a certain sunlight phase, for example, during sunrise. We use data from The Weather Channel to generate binary and nonbinary variables based on weather phenomena, including precipitation and cloud cover. Lastly, we include binary labels for weekends and holidays.

Chapter 2

The Partial Red-Blue Set Cover Problem

In this section, we introduce the *Partial Red-Blue Set Cover Problem*, a natural variant of Red-Blue Set Cover. We will show how an algorithm by Peleg [29] for Red-Blue Set Cover can be adapted to solve this new problem. In the following section, we will then explain how this problem can be used to perform exception-tolerant abduction.

2.1 Statement of the Partial Set Cover Problem

Consider a finite universe U comprised of two disjoint sets, of *red* elements R and *blue* elements B. We let β denote the number of blue elements. We suppose that we are given a collection $\mathcal{S}$ of d sets $S_1, \ldots S_d$ that are subsets of U.

For any sub-collection $\mathcal{S}' \subseteq \mathcal{S}$, let $U(\mathcal{S}')$ denote $\bigcup_{S_i \in \mathcal{S}'} S_i$, $B(\mathcal{S}')$ denote $U(\mathcal{S}') \cap B$ and $R(\mathcal{S}')$ denote $U(\mathcal{S}') \cap R$. The goal is to choose a $\mathcal{S}' \subseteq \mathcal{S}$ that covers at least μ fraction of all the elements of B while minimizing $|R(\mathcal{S}')|/|B(\mathcal{S}')|$, i.e., the number of red elements in $\mathcal{S}'$ relative to the number of blue elements.

2.2 An Algorithm for Partial Red-Blue Set Cover

We begin by defining some more useful notation. Let $deg(r_i, \mathcal{S})$ denote the number of sets in $\mathcal{S}$ that contain the red element r_i. Let $\Delta(\mathcal{S}) = \max\{deg(r_i, \mathcal{S}) : r_i \in R\}$. Denote the result of deleting elements of R' from S_i by $\phi(S_i, R') = S_i \setminus R'$ and let $\phi(\mathcal{S}, R') = \{\phi(S_i, R') : S_i \in \mathcal{S}\}$. For any set $S_i \in \mathcal{S}$ let $r(S_i) = |R(\{S_i\})|$ and for every sub-collection $\mathcal{S}' \subseteq \mathcal{S}$, let $r(\mathcal{S}') = |R(\mathcal{S}')|$. Let $H(n) = \sum_{i=1}^{n} \frac{1}{i}$ be the nth *harmonic number*.

Peleg's original algorithm used the standard greedy algorithm for approximate weighted set cover as a subroutine. Our main modification will be to replace this subroutine with a (modified) algorithm for approximate weighted *partial* set cover; Slavík established that a greedy algorithm for partial set cover achieves the same approximation ratio as for the original problem. We modify his algorithm slightly to optimize the *ratio* of the costs to number of elements covered (Algorithm 1). Precisely:

Algorithm 1 Partial Greedy Algorithm

```
Input: finite set $\mathcal{T} = \{T_1, ..., T_d\}$, costs $\{c_1, ..., c_d\}$, $\mu \in (0, 1]$
Output: $\mu$_partial cover solution set $\tilde{\mathcal{T}}$
Procedure:
  1: Set $\tilde{\mathcal{T}} = \varnothing$
  2: If $r = \mu\beta - \left|\bigcup_{t \in \tilde{\mathcal{T}}} T_t\right| \leq 0$, then STOP and output $\tilde{\mathcal{T}}$
  3: Choose the first $T_i \in \mathcal{T} \setminus \tilde{\mathcal{T}}$ that minimizes $c_t/|T_t|$, for $t \in \mathcal{T} \setminus \tilde{\mathcal{T}}$ and $T_i \neq \varnothing$.
  4: Add $T_i$ to $\tilde{\mathcal{T}}$, set $T_t = T_t \setminus T_i$, and return to step 2.
```

Theorem 1 *Let $\mathcal{T}$ be a collection of sets $T_1 \ldots, T_d$ on a universe V with corresponding weights $\omega(T_1), \ldots, \omega(T_d)$. Suppose that there is a sub-collection $\mathcal{T}^* \subseteq \mathcal{T}$ such that $T^* = \bigcup_{T \in \mathcal{T}^*} T$ contains at least $\mu|V|$ distinct elements and $\sum_{T \in \mathcal{T}^*} \omega(T) = \omega(\mathcal{T}^*)$. Then Algorithm 1 finds a subcollection $\tilde{\mathcal{T}}$ such that $\bigcup_{T \in \tilde{\mathcal{T}}} T$ also contains at least $\mu|V|$ elements and $\frac{\sum_{T_t \in \tilde{\mathcal{T}}} \omega(T_t)}{\left|\bigcup_{T_t \in \tilde{\mathcal{T}}} T_t\right|} \leq 3H(\lceil \mu|V| \rceil) \cdot \frac{\omega(\mathcal{T}^*)}{|T^*|}$.*

We now give an overview of the proof of Theorem 1, the approximation guarantee achieved by Algorithm 1. We stress that while our proof is an extension of Theorem 4 of Slavík [33], our objective is different: we are seeking to minimize the *ratio* of the cost to the size of the cover, among all covers that include a μ-fraction of the universe. Let $A = \{A_1, \ldots, A_\ell\}$ be a cover that attains this minimum value and denote by c_{min} its *cost*, $\sum_s \omega(A_s)$. We thus have, for our universe of size β, $\sum_{s=1}^{\ell} |A_s| \geq \lceil \mu\beta \rceil$. Let c_{greedy} be the cost obtained by Algorithm 1, and suppose that the cover returned includes k sets. Let $r^{(i)}$ be the number of elements remaining to be covered after the ith iteration of the algorithm, and let $A_s^{(i)}$ and $T_j^{(i)}$ denote the sets A_s and T_j after the ith iteration, i.e., after the elements from the sets $T_1, \ldots, T_i$ chosen by Algorithm 1 on all previous iterations have been removed from them.

Now, in iteration $i+1$, the greedy algorithm chooses some set j for which $c_j/|T_j^{(i)}|$ is minimized. Therefore,

$$\frac{\omega(T_{i+1}^{(i)})}{|T_{i+1}^{(i)}|} \leq \frac{\omega(A_s)}{|A_s^{(i)}|} \text{ for } s = 1, \ldots, \ell, \text{ for which } A_s^{(i)} \neq \varnothing.$$

More generally, for a given, arbitrary collection of sets $\tilde{S}$, we can define a *greedy ordering* of the sets in $\tilde{S}$, analogous to our greedy algorithm. We denote sets in the initial collection by $S_i^{(0)}$ (letting $i = 1, \ldots, |\tilde{S}|$), and put S_1 equal to some first set minimizing the ratio $\frac{\omega(S)}{|S|}$. Then, given inductively that we have chosen the ordering up to j, we put each $S_i^{(j)} = S_i^{(j-1)} \setminus S_j$, i.e., equal to the elements of $S_i^{(0)}$ that are still uncovered by the partial collection up to j, and take S_{j+1} to be the first set minimizing the ratio $\frac{\omega(S_{j+1})}{|S_{j+1}^{(j)}|}$.

We observe that the final ratio achieved by the collection $\tilde{S}$ is a weighted average of these ratios:

$$\frac{\sum_j \omega(S_j)}{\left|\bigcup_j S_j\right|} = \sum_i \frac{|S_{i+1}^{(i)}|}{\left|\bigcup_j S_j\right|} \frac{\omega(S_{i+1})}{|S_{i+1}^{(i)}|}$$

where $\sum_i \frac{|S_{i+1}^{(i)}|}{|\bigcup_j S_j|} = 1$. We also observe that for $j < k$, $\frac{\omega(S_i)}{|S_i^{(j)}|} < \frac{\omega(S_i)}{|S_i^{(k)}|}$.

Lemma 2 *There is an optimal cover in which only the final set in any greedy ordering may contain more than $\mu\beta$ elements, and the collection of all prior sets covers fewer than $\mu\beta$ elements.*

Proof: Consider the greedy ordering of the sets in any optimal cover. Since for $j < k$, $\frac{\omega(A_i)}{|A_i^{(j)}|} \leq \frac{\omega(A_i)}{|A_i^{(k)}|}$, if a set is not chosen before the first set by which $\mu\beta$ elements have been covered, then its ratio will always be at least as large as that of the set for the index at which the given cover contains $\mu\beta$ elemtns, as well as all previous sets in the ordering. Since the overall ratio achieved by the cover is an averaging of these ratios, eliminating sets that appear in the ordering after this point can only improve the ratio achieved by the cover. ■

Now consider all fractions of the form $\frac{\omega(A_s)}{k_s}$ for $s = 1, \ldots, \ell$ and $k_s = 1, \ldots, |A_s|$. Note that there are at least $\lceil \mu\beta \rceil = r^{(0)}$ such fractions. Suppose we arrange these fractions into a nonincreasing sequence $e_1 \geq e_2 \geq \cdots \geq e_{r^{(0)}} \geq \cdots$. Closely following Slavík, we then obtain the following inequalities.

Lemma 3 (c.f. Lemma 1 of Slavík [33]) *For* $i = 0, \ldots, k-1$, $\frac{\omega(T_{i+1})}{|T_j^{(i)}|} \leq e_{r^{(i)}}$.

Proof: After i^{th} iteration of the greedy algorithm, $i = 0, ..., k-1$, there are exactly $r^{(i)}$ elements to be covered, i.e. there are at least $r^{(i)}$ elements in $\cup_{s=t}^{l} a_s^{(i)} \leq r^{(i)}$.
The greedy condition implies that
$\omega(T_{i+1})/u_{i+1}^{(i)} \leq \omega(A_s)/a_s^{(i)}$ for all s=1,...,l for which $a_s^{(i)} > 0$. Therefore
$\omega(T_{i+1})/u_{i+1}^{(i)} \leq \omega(A_s)/k_s forall s = 1, ..., l$ and all $k_s = 1, ..., a_s^{(i)}$, i.e.
$\omega(T_{i+1})/u_{i+1}^{(i)} \leq e_j$ for at least $r^{(i)}$ indices j, but $e_1 \geq ... \geq e_{r^{(0)}}$ hence
$\omega(T_{i+1})/u_{i+1}^{(i)} \leq e_{r^{(i)}}$ for any $i = 0, ..., k-1$

■

Lemma 4 (c.f. Lemma 2 of Slavík [33]) $c_{greedy} \leq \sum_{s=1}^{\ell-1} \omega(A_s) H(|A_s|) + \omega(A_\ell) H(\min\{\lceil \mu\beta \rceil, |A_\ell|\})$

Proof: $\frac{\omega(T_{i+1})}{u_{i+1}^{(i)}} \leq e_{r^{(i)}} \leq e_{r^{(i)}-1} \leq ... \leq e_{r^{(i+1)}+1}$. Since $r^{(i)} - r^{(i+1)} = u_{i+1}^{(i)}$, we have
$\omega(T_{l+1}) \leq e_{r^{(i)}} + e_{r^{(i)}-1} + ... + e_{r^{(i+1)}+1}$.
Adding the above inequalities for $i = 0...k-1$, we have
$\omega(T_1) + ... + \omega(T_k) \leq e_{r^{(0)}} + ... + e_l = \sum_{s=1}^{\ell-1} \omega(A_s) H(|A_s|) + \omega(A_\ell) H(\min\{\lceil \mu\beta \rceil, |A_\ell|\})$ ■
We now handle the special case in which every optimal cover is dominated by a single, large set:

Lemma 5 *If every optimal cover contains more than* $3\lceil \mu\beta \rceil$ *elements and is more than three times the size of the greedy algorithm's cover, then the greedy algorithm achieves an approximation ratio of* 3.

Proof: We observe that in the first case, by Lemma 2, the optimal ratio is at least $\frac{2}{3}\frac{\omega(A_\ell)}{|A_\ell^{(0)}|}$ since the final set contributes at least 2/3 of the final ratio. Furthermore, since the greedy algorithm's cover contains at most 1/2 of $A_\ell^{(0)}$, we know that the ratios of the sets selected by the greedy algorithm are all at most $2\frac{\omega(A_\ell)}{|A_\ell^{(0)}|}$. Since, again, the greedy algorithm's ratio is a weighted average of these individual ratios, the final ratio is also at most $2\frac{\omega(A_\ell)}{|A_\ell^{(0)}|}$, and hence at most three times larger than the optmal ratio. ■

Finally, we can prove Theorem 1

Proof of Theorem 1: Fix a smallest optimal cover. Lemma 2 implies that $|A_s| \leq \lceil \mu\beta \rceil$ for all $s = 1, \ldots, \ell - 1$. This, and Lemma 4 establish that $c_{greedy} \leq H(\lceil \mu\beta \rceil)c_{min}$. Now, if the optimal cover contains fewer than $3\lceil \mu\beta \rceil$ elements, since the greedy algorithm must return a cover with at least $\lceil \mu\beta \rceil$ elements, a ratio of $3H(\lceil \mu\beta \rceil)$ is immediate in this case. More generally, if the optimal cover contains at most three times as many elements as the cover returned by the greedy algorithm the ratio is again $3H(\lceil \mu\beta \rceil)$. Finally, Lemma 5 guarantees that if neither of these cases hold, we still obtain a ratio of $3 \leq 3H(\lceil \mu\beta \rceil)$. ■

We sketch the proof of Theorem 1 as above. Now, we modify Peleg's subroutine GREEDY_RB to use Algorithm 1 instead of the standard greedy algorithm, obtaining Algorithm 2.

Algorithm 2 Greedy partial RB

```
Input: finite set $\mathcal{S} = \{S_1, ..., S_d\}$, $\mu \in (0, 1]$
Output: $\mu$_partial cover solution $\tilde{\mathcal{S}}$
Procedure:
1: Modify $\mathcal{S}$ into an instance $\mathcal{T}$ of the weighted set cover problem as follows: (a) Take
   $\mathcal{T} = \phi(\mathcal{S}, R)$ (b) Assign each set $T_i = \phi(S_i, R)$ in $\mathcal{T}$ a weight $\omega(T_i) = r(S_i)$
2: Apply Algorithm 1 for weighted partial set cover to $\mathcal{T}$ and generate a cover $\tilde{\mathcal{T}}$
3: Get the corresponding collections as a set of solutions $\tilde{\mathcal{S}} = \{S_i : T_i \in \tilde{\mathcal{T}}\}$
```

Lemma 6 *Algorithm Greedy partial RB yields an approximation ratio of* $\Delta(\mathcal{S}) \cdot 3H(\mu\beta)$

Meanwhile, We will use the following lemma from Peleg:

Lemma 7 *[29, Lemma 3.1] For any collection* $\mathcal{S}' \subseteq \mathcal{S}$ *and the corresponding instance* $\mathcal{T}' = \phi(\mathcal{S}', R)$ *of the weighted set cover problem* $r(\mathcal{S}') \leq \omega(\mathcal{T}') \leq \Delta(\mathcal{S}) \cdot r(\mathcal{S}')$

The proof is now very similar to that of the analogous lemma, Lemma 3.2 used by Peleg:

Proof of Lemma 6: Let any minimum-weight set cover $\mathcal{T}'$ of $\mathcal{T}$ be given. Consider any optimal cover $\mathcal{S}^* \subseteq \mathcal{S}$ that covers $\mu\beta$ blue elements and put $\mathcal{T}^* = \phi(\mathcal{S}^*, R)$. Since, by Theorem 1, Algorithm 1 yields a $3H(\mu\beta)$ approximation ratio for the weighted partial set cover problem, we then have that the solution returned by Algorithm 1 satisfies $\omega(\tilde{\mathcal{T}}) \leq 3H(\mu\beta) \cdot \omega(\tilde{\mathcal{T}}')$. Lemma 7 then gives $r(\tilde{\mathcal{S}}) \leq \omega(\tilde{\mathcal{T}})$. And, since $\tilde{\mathcal{T}}'$ is an optimal partial cover of $\mathcal{T}$, $\omega(\tilde{\mathcal{T}}') \leq \omega(\tilde{\mathcal{T}}^*)$. In summary, so far we have

$$r(\tilde{\mathcal{S}}) \leq \omega(\tilde{\mathcal{T}}) \leq 3H(\mu\beta) \cdot \omega(\tilde{\mathcal{T}}') \leq 3H(\mu\beta) \cdot \omega(\tilde{\mathcal{T}}^*).$$

Now, Lemma 7 gives

$$3H(\mu\beta) \cdot \omega(\tilde{\mathcal{T}}^*) \leq 3H(\mu\beta) \cdot \Delta(\mathcal{S}) \cdot r(\tilde{\mathcal{S}}^*)$$

completing the proof. ■

We next modify the body of Peleg's main algorithm, Low_Deg in the natural way to obtain our final algorithm, (1) replacing the use of Greedy_RB with Algorithm 2, (2) checking that the family may possibly admit a sufficiently large partial covering after computing $\mathcal{S}_X$, and (3) computing our notion of (relative) error rate rather than just the number of red elements.

Algorithm 3 Low Deg Partial(X)

Input: finite set $\mathcal{S} = \{S_1, ..., S_d\}$, $\mu \in (0, 1]$, integer X
Output: μ_partial cover solution $\tilde{\mathcal{S}}_X$ and corresponding error rate $\tilde{\epsilon}$
Procedure:

```
1: Discard sets in $\mathcal{S}$ that contain more than $X$ red elements, set $\mathcal{S}_X \leftarrow \{S_i \in \mathcal{S} : r(S_i) \leq X\}$.
2: If $\frac{|B(\mathcal{S}_X)|}{|B|} < \mu$, then return FAIL          $\triangleright$ $\mathcal{S}_X$ is not feasible
3: Set $Y = \sqrt{\frac{d}{H(\lceil \mu\beta \rceil)}}$
4: Identify the high degree red elements: $R_H \leftarrow \{r_i \in R : deg(r_i, \mathcal{S}_X) > Y\}$
5: Discard elements of $R_H$ in $\mathcal{S}_X$: $\mathcal{S}_{X,Y} \leftarrow \phi(\mathcal{S}_X, R_X)$
6: Apply Algorithm 2 to $\mathcal{S}_{X,Y}$ and obtain a solution $\tilde{\mathcal{S}}_{X,Y}$ for it.
7: Add the dropped red elements back to obtain the corresponding result $\tilde{\mathcal{S}}_X$.
8: For the set of blue elements $\tilde{B}$ and red elements $\tilde{R}$ respectively covered by $\tilde{\mathcal{S}}_X$, calculate
   the error rate $\tilde{\epsilon} = \frac{|\tilde{R}|}{|\tilde{B}|}$ and return it and $\tilde{\mathcal{S}}_X$.
```

Algorithm 4 Low Deg Partial 2

Input: finite set $\mathcal{S} = \{S_1, ..., S_d\}$, $\mu \in (0, 1]$
Output: optimal choice of μ_partial cover solution and corresponding error rate $\hat{\epsilon}$
Procedure:

```
1: For X=1 to $|R|$ do:
2:     Low Deg Partial(X)
3: Take the solution that yields the lowest error rate
```

Theorem 8 *Algorithm 4 solves the Partial Red-Blue Set Cover problem with an approximation ratio of* $4\sqrt{d \cdot H(\mu\beta)}$

The proof of Theorem 8 is virtually identical to the proof of Theorem 3.5 of Peleg (and the proof Peleg's Lemma 3.4), with our Lemma 6 replacing Lemma 3.2, and $3H(\mu\beta)$ replacing the original $\log \beta$ approximation ratio.For convenience, we still show the proof below:

Proof: Each set $S_i \in S_X$ has at most X red elements, hence $|R_H| {*} Y \leq \sum_{r_{j \in R_H}} deg(r_j, S_X) \leq \sum_{r_{j \in R}} deg(r_j, S_X) = \sum_{S_i \in S_X} r(S_i)|S_X| {*} XdX$ so $|R_H| \leq dX$ implying that $R_H \leq \sqrt{dH(\mu\beta)} {*} X$.
Next, we define $\hat{X}$ as our optimal value. First, $S_{\hat{X}}$ is necessarily feasible. Hence the procedure will always return a solution in step 2 in Algorithm 3. Let S^* be some optimal solution for the problem and let $r_H^* = |R(S^*) \cap R_H|$ and $r_L^* = |R(S^*) \cap R_L|$. Since $\Delta(S_{\hat{X},Y}) \leq Y$, Lemma 6 guarantees that the solution produced by algorithm 3 uses at most $Y * 3H(\mu\beta) * r_L^* = 3\sqrt{dH(\mu\beta)} * r_L^*$ red elements of R_L^* and $R_H \leq \sqrt{dH(\mu\beta)} * \hat{X}$. Combined, the total number of red elements used by the procedure satisfies $r(\tilde{SX}) \leq 3\sqrt{dH(\mu\beta)} r_L^* + \sqrt{dH(\mu\beta)} * \hat{X}$. But by the definition of $\hat{X}$, necessarily $r(S^*) \geq \hat{X}$ and hence $r(\tilde{SX}) \leq 4\sqrt{dH(\mu\beta)} r(S^*)$ yielding the Theorem. ■

Chapter 3

Using Partial Red-Blue Set Cover Algorithms for Exception-Tolerant Abduction

We will now show that, given an appropriate number of examples, algorithms for the Partial Red-Blue Set Cover problem can be used to perform exception-tolerant abduction. We first recall the PAC-learning formulation of abduction proposed by Juba [17].

3.1 Learning of Exception-Tolerant k-DNF Abduction

The formulation of learning exception-tolerant abduction is as follows. Suppose there are n propositional attributes $x_1, \ldots, x_n$, and we are given a *query* to be explained, a Boolean formula c that may use our n propositional attributes as variables. We fix an alphabet $A \subseteq \{x_1, \ldots, x_n\}$ of attributes we wish to allow in our explanations. For example, A may only contain the attributes that take values "before" the attributes used in the formula c describing the event to be explained. We are also given as input m examples, $x^{(1)}, \ldots, x^{(m)}$, drawn independently from a common, unknown distribution D over Boolean values for all of the n attributes. We are given a target *plausibility* threshold $\mu \in (0, 1)$, and an integer k for the complexity of our solutions. Following Juba [17], we will only seek to use k-DNFs as explanations; it seems that this is essentially the most expressive natural class of formulas for which this task is tractable. Finally, we fix a *tolerance* $\gamma \in (0, 1/3]$ indicating the amount of loss relative to the optimal plausibility we are willing to accept. Let

$$\epsilon^* = \min_{k\text{-DNF } h \text{ on } A: \Pr[h(x)=1] \geq \mu} \Pr[c(x) = 0 | h(x) = 1]$$

be the optimal error rate achievable by a k-DNF using only attributes in A that is satisfied at least a μ-fraction of the time on D.

Our task is now to return a k-DNF h that uses only attributes in A such that with probability $1 - \delta$ over the draw of $x^{(1)}, \ldots, x^{(m)}$ from D,

1. *Plausibility.* $\Pr[h(x) = 1] \geq (1 - \gamma)\mu$ and
2. *Entailment.* $\Pr[c(x) = 0 | h(x) = 1] \leq \alpha(n, 1/\epsilon^*, 1/\delta)\epsilon^*$ where we say that $\alpha(n, 1/\epsilon^*, 1/\delta)$ is the *approximation ratio* achieved by our algorithm.

Note that we are seeking to learn *both* the (approximate) entailment relation between the various hypotheses and the conclusion c and the degree of plausibility of the various hypotheses from the examples.

Our task is formally equivalent to finding a prediction rule h for c that achieves a *positive classification rate* of $(1 - \gamma)\mu$ and *precision* $1 - \alpha\epsilon^*$, given that some other unknown rule h^* with a positive classification rate μ achieves precision $1 - \epsilon^*$.[2]

3.2 Analysis of Partial Red-Blue Set Cover Algorithms for Learning Abduction

We will now prove our main theorem, stating that Algorithm 4 can be used to perform exception-tolerant abduction with an approximation ratio of $O(\sqrt{n^k \log \mu m})$, where $m = \Theta(\frac{1}{\gamma^2 \mu \epsilon^*}(n^k + \log \frac{1}{\delta}))$ (in particular the factor of μ in the approximation ratio cancels the factor of $1/\mu$ in m).

[2]Our algorithm can be easily extended to achieving *recall* $(1 - \gamma)\mu$ and precision $1 - \alpha\epsilon^*$ when a rule achieving recall μ and precision $1 - \epsilon^*$ exists—one simply *only creates a blue element for the positive examples* instead of all examples.

Theorem 9 *Suppose we are given* $m = \Theta(\frac{1}{\gamma^2\mu\epsilon^*}(n^k + \log\frac{1}{\delta}))$ *examples. Then Algorithm 4 can be used to solve the exception-tolerant abduction task in time polynomial in* m *and* n^k *with approximation ratio* $O(\sqrt{n^k \log \mu m}) = O(\sqrt{n^k \log \frac{n+\log 1/\delta}{\gamma\epsilon^*}})$.

To prove this theorem, we will need to argue that the "empirical" problem posed by a fixed training set provides a good approximation to the quality of a k-DNF explanation on the actual distribution of examples.

Lemma 10 *For any* $c : \{0,1\}^n \to \{0,1\}$, $\delta \in (0,1)$, *and* $\gamma \in (0, 1/3]$, *let* $x^{(1)}, \ldots, x^{(m)}$ *be independently drawn from a common distribution* D *over* $\{0,1\}^n$ *for*

$$m \geq \frac{3(1+\gamma)}{\gamma^2(1-\gamma)\mu\epsilon^*}(\ln 2\binom{2n}{k} + \ln\frac{4}{\delta})$$

where ϵ^* *is the minimum (nonzero)* $\Pr[c(x) = 0|h(x) = 1]$ *over* k*-DNFs* h *with* $\Pr[h(x) = 1] \geq \mu$ *for a given target* μ.

Then with probability $1-\delta$ *over the draw of* $x^{(1)}, \ldots, x^{(m)}$, *if a* k*-DNF* h *is true on a* $\hat{\mu}$ *fraction of* $x^{(1)}, \ldots, x^{(m)}$ *for* $\hat{\mu} \geq (1-\gamma)\mu$, *we have*

$$(1+\gamma)\Pr[h(x) = 1] \geq \hat{\mu} \geq (1-\gamma)\Pr[h(x) = 1].$$

If, furthermore $\Pr[c(x) = 0|h(x) = 1] \geq \epsilon^*$ *and* $c(x^{(j)}) = 0$ *for a* $\hat{\epsilon}$ *fraction of* $\{x^{(j)} : h(x^{(j)}) = 1\}$, *we have*

$$\begin{aligned}(1-2\gamma)\Pr[c(x) = 0|h(x) = 1] &\leq \hat{\epsilon}\\ &\leq (1+3\gamma)\Pr[c(x) = 0|h(x) = 1].\end{aligned}$$

So in short, for every k-DNF h, one of three cases hold: either h is satisfied on too few examples to be considered (fewer than $(1-\gamma)\mu$), or h has error better than our target optimum ϵ^* (over those h' satisfied with probability at least μ), or else we have good estimates of the error made by h at justifying c.

This lemma is a straightforward consequence of the (multiplicative) Chernoff bound:

Theorem 11 (Multiplicative Chernoff bound) *Let $X_1, \ldots, X_m$ be independent random variables taking values in $[0,1]$, such that $\mathbb{E}[\frac{1}{m}\sum_i X_i] = p$. Then for $\gamma \in [0,1]$,*

$$\Pr\left[\frac{1}{m}\sum_i X_i > (1+\gamma)p\right] \le e^{-mp\gamma^2/3}$$
$$\text{and } \Pr\left[\frac{1}{m}\sum_i X_i < (1-\gamma)p\right] \le e^{-mp\gamma^2/2}$$

Proof of Lemma 10: Let any c and k-DNF h be given. We will use the Chernoff bound to bound the probability that the sample substantially misrepresents either the probability that h is satisfied or the probability of h failing to entail c.

First, we observe that $h(x^{(1)}), \ldots, h(x^{(m)})$ indeed take values in $[0,1]$. We will let $p(h)$ denote $\mathbb{E}[h(x^{(j)})] = \Pr[h(x) = 1]$. A first application of the Chernoff bound guarantees that for this h,

$$\Pr[\hat{\mu} > (1+\gamma)p(h)] \le e^{-mp(h)\gamma^2/3} \text{ and}$$
$$\Pr[\hat{\mu} < (1-\gamma)p(h)] \le e^{-mp(h)\gamma^2/2}.$$

We also note that there are $2^{\binom{2n}{k}}$ k-DNFs. Thus we find by a union bound over all of the k-DNF that the probability of these bounds failing is $2 \cdot 2^{\binom{2n}{k}} \cdot e^{-mp(h)\gamma^2/3}$. So in particular, with probability $1-\delta/2$, any h with $p(h) < \frac{1-\gamma}{1+\gamma}\mu$ has $\hat{\mu} < (1-\gamma)\mu$, and otherwise, $\hat{\mu}$ is a suitable estimate of $p(h)$.

Likewise, the indicator functions $I[h(x^{(j)}) = 1 \wedge c(x^{(j)}) = 0]$ also take values in $[0,1]$ and we will let $\varepsilon(h)$ denote

$$\mathbb{E}[I[h(x^{(j)}) = 1 \wedge c(x^{(j)}) = 0]] = \Pr[h(x) = 1 \wedge c(x) = 0].$$

We also note that $\hat{\epsilon} = \frac{1}{\hat{\mu}}\sum_{j=1}^{m} I[h(x^{(j)}) = 1 \wedge c(x^{(j)}) = 0]$. So, a second application of the Chernoff bound guarantees that h also satisfies

$$\Pr[\hat{\mu}\hat{\epsilon} > (1+\gamma)\varepsilon(h)] \le e^{-m\varepsilon(h)\gamma^2/3} \text{ and}$$
$$\Pr[\hat{\mu}\hat{\epsilon} < (1-\gamma)\varepsilon(h)] \le e^{-m\varepsilon(h)\gamma^2/2}.$$

Now, we note that for h with $p(h) > \frac{1-\gamma}{1+\gamma}\mu$, either $\Pr[c(x) = 0|h(x) = 1] \leq \epsilon^*$ or else $\varepsilon(h) \geq \frac{1-\gamma}{1+\gamma}\mu\epsilon^*$ Thus, by another union bound over these two inequalities and all suitable k-DNFs, the probability of any of these bounds failing is at most $2 \cdot 2^{\binom{2n}{k}} \cdot e^{-m\frac{1-\gamma}{1+\gamma}\mu\epsilon^*\gamma^2/3}$. Thus, now, for the claimed m, with probability $1-\delta$ all of these bounds simultaneously hold, and we additionally get

$$\frac{1-\gamma}{1+\gamma}\frac{\varepsilon(h)}{p(h)} \leq \hat{\epsilon} \leq \frac{1+\gamma}{1-\gamma}\frac{\varepsilon(h)}{p(h)}.$$

Of course, $\varepsilon(h)/p(h) = \Pr_{x\in D}[c(x) = 0|h(x) = 1]$ and $\frac{1+\gamma}{1-\gamma} \leq 1 + 3\gamma$ since $\gamma < 1/3$. ■

We are now ready to prove our main theorem.

Proof of Theorem 9: We produce the following instance of Partial Red-Blue Set Cover: we create a blue element for each example $x^{(1)}, \ldots, x^{(m)}$, create a red element for each example $x^{(j)}$ such that $c(x^{(j)}) = 0$, and create a set for each term of size k using attributes in A containing each blue element such that the corresponding $x^{(j)}$ satisfies that term. Let $\hat{\epsilon}^*$ be the smallest fraction of red elements covered by any family of these sets that covers at least $(1 - \gamma/2)\mu m$ blue elements (i.e., examples). Note that there are m blue elements and $\binom{2|A|}{k}$ sets.

Theorem 8 then establishes that Algorithm 4 run on this instance with parameter $(1-\gamma/2)\mu$ returns a set S of terms of size k using attributes in A such that:

1. $\hat{\mu}m \geq (1 - \gamma/2)\mu m$ elements are satisfied by some term in S.
2. The number of $x^{(j)}$ such that $c(x^{(j)}) = 0$ that are satisfied by any term in S is at most $4\sqrt{\binom{2|A|}{k}H(\mu m)\hat{\epsilon}^*\hat{\mu}m}$.

Consider any k-DNF h^* with $\Pr[h^*(x) = 1] \geq \mu$ that achieves $\Pr[c(x) = 0|h^*(x) = 0] = \epsilon^*$. Lemma 10 now guarantees that if we use $\gamma/2$ as our tolerance parameter, h^* is satisfied on at least $(1 - \gamma/2)\mu$ examples, and at most $(1 + 3\gamma/2)\epsilon^*$ examples that satisfy h^* also have $c(x^{(j)}) = 0$. Therefore, $\hat{\epsilon}^* \leq (1 + 3\gamma/2)\epsilon^*$, and Algorithm 4 must find a family of sets corresponding to a k-DNF h such that at most a $4(1 + 3\gamma/2)\sqrt{\binom{2|A|}{k}H(\mu m)\epsilon^*}$ fraction of examples satisfy h but not c.

Now, since Algorithm 4 must return a k-DNF h that satisfies at least $(1-\gamma/2)\mu m$ examples, Lemma 10 also guarantees that actually with probability $1-\delta$, $\Pr[h(x) = 1] \geq (1-\gamma/2)^2\mu \geq$

$(1-\gamma)\mu$ and, using the fact that $\gamma \leq 1/3$ and the standard bounds $H(x) \leq 1 + \ln x$ and $\binom{n}{k} \leq \left(\frac{ne}{k}\right)^k$, where e is the base of the natural logarithm,

$$\begin{aligned}\Pr[c(x)=0|h(x)=1] &\leq (2+3\gamma)^2 \sqrt{\binom{2|A|}{k} H(\mu m)\epsilon^*} \\ &\leq 9\sqrt{\left(\frac{2en}{k}\right)^k (1+\ln \mu m)\epsilon^*}.\end{aligned}$$

We can further bound this expression by using that $m = \Theta\left(\frac{1}{\gamma^2\mu\epsilon^*}(\binom{2n}{k} + \log\frac{1}{\delta})\right)$. We thus find that it is $O(\sqrt{n^k \log \frac{n+\log 1/\delta}{\gamma\epsilon^*}}\epsilon^*)$ as claimed. We find furthermore by inspection that the algorithm indeed runs in time polynomial in m and n^k since all of the parameters – the number of red and blue elements, the number of sets, and the degree of each element – can be bounded by such polynomials. ■

3.3 A Toy Example

To better understand the algorithm, we now consider an example. Suppose that we have the following set of examples:

Event #	Wet Clothes	Raining	Sleep Well	Inside
1	yes	yes	no	no
2	no	no	yes	no
3	yes	yes	yes	no
4	no	yes	yes	yes
5	no	yes	yes	yes
6	yes	yes	no	no
7	no	no	yes	yes

Suppose that we want to propose a reason that clothes become wet. We can translate this small data set into a Partial Red-Blue Set Cover problem as shown in Figure 3.1.

In particular, suppose that we are in Algorithm 4, with error tolerance $X = 2$ and $\mu = 6/7$. Then in Algorithm 3, first in Step 3.1, we will discard sets in $\mathcal{S}$ that contains more than $X = 2$

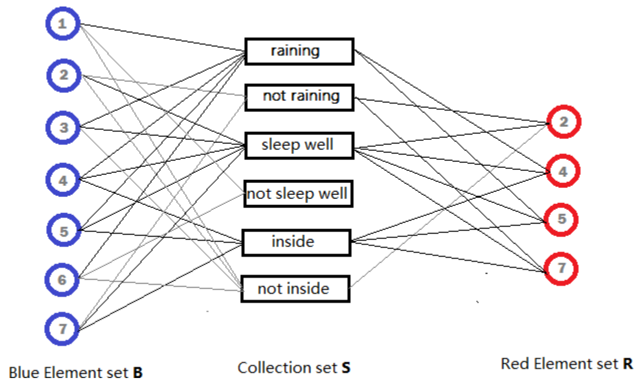


Figure 3.1: Red-Blue Set Cover instance for the condition 'wet clothes' in our example data set. We draw an edge joining a set and an element if the set contains that element. Every example has a blue element, and the examples where 'wet clothes' = no have a red element.

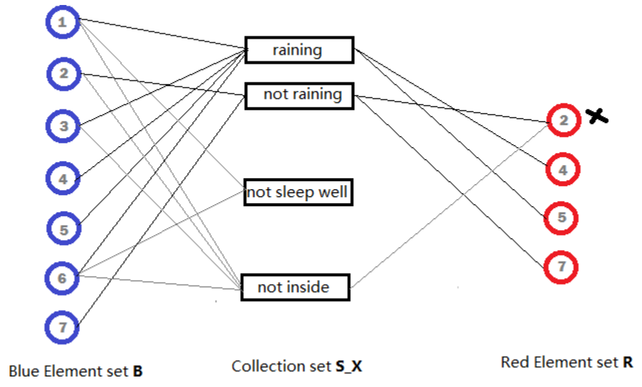


Figure 3.2: The Red-Blue Set Cover instance after the sets containing more than $X = 2$ red elements are discarded. Note that the red element #2 is contained in $2 > Y \approx 1.3$ sets, and will also be discarded.

red elements. Thus we obtain $\mathcal{S}_X = \{$'raining', 'not raining', 'not sleep well', 'not inside'$\}$, as illustrated in Figure 3.2. Notice that this removes the connection between red elements and discarded collections. Next, in Step 3.2, we check if the whole set $\mathcal{S}_X$ contains enough blue elements for our objective value μ. This $\mathcal{S}_X$ indeed contain enough blue elements. The "degree bound" Y calculated in Step 3.3 is $Y \approx 1.3$. For Step 3.4, we will identify the "high degree" red elements and create another set $\mathcal{S}_{X,Y}$ that does't have these high degree red elements. For this example, red element #2 is considered to be a high degree red element. Intuitively, we drop these high degree elements because we consider it likely that we will end up including these points sooner or later, so we don't want to penalize sets for containing them. We only want to "charge" a set for the "unusual" (low degree) red elements it contains.

Finally we run the greedy algorithm on $\mathcal{S}_{X,Y}$ to obtain a solution $\tilde{\mathcal{S}}_{X,Y}$. We then add back the dropped red elements. The result, $\tilde{\mathcal{S}}_X$ might be $\{$'not inside', 'raining'$\}$, corresponding to the 1-DNF 'not inside' $\vee$ 'raining'. It covers blue elements $\{1, 2, 3, 4, 5, 6\}$, i.e., it is satisfied on the corresponding examples, and its error rate is 0.5.

Chapter 4

Evaluation for the Algorithm

4.1 Empirical Evaluation

So far, we have proposed a new algorithm for exception-tolerant abduction and proved a better worst-case approximation guarantee for this algorithm than was known for the Tolerant Elimination algorithm proposed for this task by Juba [17]. Although such worst-case guarantees are desirable, they do not rule out the possibility that Tolerant Elimination might still obtain results as good as or better than our new algorithm on various actual distributions. So, we have investigated the performance of the two algorithms on a couple simple synthetic domains.

The first domain is an example of an "ideal" situation for our algorithms: here, the target condition c is generated by a hidden k-DNF that has been corrupted by some independent random noise. Ideally, the algorithms should obtain a hypothesis that is satisfied with approximately the same probability (less the noise) that the hidden rule would be satisfied, and with an error rate that is approximately the noise rate. The second domain is an example of the challenging situation that we hope our algorithms can cope with. The target condition c is generated by a random *linear threshold function*, i.e., a random (centered) half-space of the Boolean cube. k-DNF formulas cannot approximate such rules well,[3] so we can only hope to obtain a low error rate by choosing a hypothesis that is satisfied relatively rarely. That is, this is a domain in which the "errors" are highly regular, but the rule we wish to explain

[3]This is not obvious, but O'Donnell and Wimmer obtain such a result for the simple majority function [27], where our threshold functions are a random rotation, which have similar *"influences"* and are similarly hard for k-DNFs to approximate. See O'Donnell [26, Chapters 4–5] for more.

is simply too complex for the representations we use. It therefore tests the capacity for our algorithms to propose a reasonable hypothesis under relatively unfavorable circumstances.

In the second domain, we also tested a simpler greedy covering algorithm that orders the terms by their empirical error rates, and simply adds terms to the k-DNF until it has covered the desired empirical fraction of the data. This method is intended as a baseline. It does not feature the same theoretical guarantees as our new algorithm.

4.1.1 Noisy Planted k-DNF

Here, we first choose a k-DNF of a fixed size $s(k)$ by selecting $s(k)$ terms uniformly at random (with replacement) from the terms of size k. $s(k)$ was selected to be relatively large while keeping the probability of the k-DNF being satisfied around 99%, so that we can sample both satisfying and falsifying assignments relatively easily: here, we take $s(1) = 6$ and $s(2) = 16$. Once this "planted" k-DNF φ is fixed, we take the distribution D to generate a uniform satisfying assignment of φ with probability .15, and a uniform falsifying assignment of φ with probability .85. We can sample from D using simple rejection sampling: we draw a uniform random example, and if it satisfies φ, we independently restart (rejecting the example) with probability $\alpha(\varphi)$ so that we obtain the desired ratio of satisfying and falsifying examples. In our experiments, we used 100 attributes and generated 10 formulas for each k. We then generated 50,000 examples for each formula, a typical size training set.

For each example x, we independently chose whether to put $c(x) = \varphi(x)$ with 95% probability, or to put $c(x) \neq \varphi(x)$ with 5% probability. That is, there is a *noise rate* of 5%. So, we know that the hidden φ agrees with c except on a random $\approx 5\%$ of examples. Therefore φ itself, at least, is a k-DNF that (1) explains approximately a $0.95 \cdot 0.15$ fraction of examples drawn from this distribution and (2) c only fails to hold on 5% of the examples drawn from this distribution on which φ is satisfied. We supplied these labeled examples to each of the algorithms, and to estimate the error of the hypothesis the algorithms produced, drew another data set using the same planted φ and computed the error on this new data set. We repeated this process 10 times each with independently sampled φ, to get an estimate of the distribution of error rates for these algorithms. We supplied Tolerant Elimination the actual noise parameter of 5% and we supplied the Low-Degree algorithm with the actual fraction, 14.25%, of the data that we expect the planted k-DNF to explain. The results (empirical mean and standard deviation) were as shown below:

	1-DNF error	2-DNF error
Tol. Elim.	$5.985\% \pm 0.341\%$	$5.915\% \pm 0.304\%$
Low Deg.	$5.996\% \pm 0.353\%$	$5.847\% \pm 0.371\%$

We see that both algorithms succeeded at this simple task, matching the error rate of the planted k-DNF.

4.1.2 Random Linear Threshold Rules

For this experiment, we first choose a random linear threshold rule as follows: we first generate a weight vector θ in which each coordinate is drawn independently from a centered binomial distribution with parameters $(100, 1/2)$. (We take this as an approximation of a scaling of a multivariate Gaussian distribution with mean 0.) We take $n = 100$ attributes *except* for our 3-DNF experiments, where we take $n = 20$ attributes (on account of the exploding number of terms of size 3). For each example x in $\{0,1\}^n$, we convert $x \in \{0,1\}$ to $y \in \{-1,1\}$ (e.g., using $y_i = 2x_i - 1$); we then put $c(x)$ equal to $[\langle \theta, y \rangle \geq 0]$, i.e., 1 if the inequality holds and 0 otherwise. We generated 10 different linear threshold rules from this distribution, and generated 10,000 examples uniformly at random for 1-DNF and 2-DNF, and 6,000 examples for 3-DNF.

We ran the algorithms on these training sets, giving the low-degree and naive greedy (baseline) algorithms $\mu = 10\%, 30\%, 50\%, 70\%, 90\%$ and 100%, and giving tolerant elimination a variety of different target error rates; only $\epsilon = 16\%$ for 2-DNF had any nontrivial effect. We then generated 7,500 additional uniform random examples for 1-DNF and 2-DNF to serve as a test set, and 4,500 examples for 3-DNF. We evaluated the quality of the hypothesis produced for each training set on these test sets; the results of this evaluation appear in Figure 4.1.

We make three observations about the results of this experiment. First, the results illuminate a striking weakness of the Tolerant Elimination algorithm. The algorithm is forced to pick a threshold error rate that it uses to select whether or not to include a term in its hypothesis.

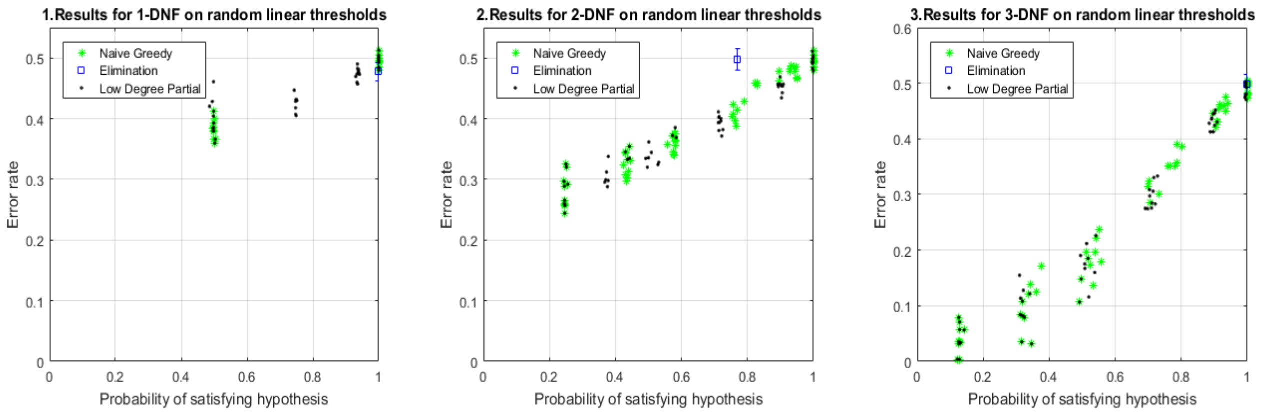


Figure 4.1: Experiments with random linear threshold rules. We chose a random linear threshold centered in the Boolean cube that selects approximately half of the points, so an error rate of 50% is trivial. We plot the mean and standard deviation of the error rate for tolerant elimination. Both the new low degree partial cover algorithm and our naive greedy baseline obtain hypotheses with significantly low error rates, whereas the original tolerant elimination algorithm cannot obtain a nontrivial error rate. The new algorithm also obtains slightly lower error rates (covering closer to the target fraction) than the naive baseline.

While this works relatively well in the noisy k-DNF setting where the terms with lower error rates are terms of the hidden k-DNF, it fails badly here, forcing the k-DNF to pick many or few terms. For example, the best we can do for 1-DNF is essentially to use the literal corresponding to the largest weight component of the linear threshold rule. (This is what both the low-degree and naive greedy algorithms produce to explain 50% of the data.) But, there are *many* literals with essentially similar weights, and each additional literal that is selected, the hypothesis picks up half of the remaining possible examples. It is very difficult to discover an "ideal" setting for the tolerance, and in our experiments the algorithm always selected a hypothesis that was not substantially better than the trivial hypothesis that is always satisfied—both achieved error rates of $\approx 50\%$.

The second observation is that by contrast, both our low-degree partial cover algorithm and the naive greedy baseline algorithm obtained significantly lower error rates. That is, both algorithms were reliably able to successfully infer nontrivial rules in this challenging domain. In general, we obtained (as one would expect) a trade-off between the probability that the generated hypothesis was satisfied and its error rate. Also, from $k = 1$ to $k = 2$, the error rate we obtain for the same fraction decreases (but note that the data we used for 3-DNF had far fewer attributes, and hence is inherently easier to approximate and is not comparable to $k = 1, 2$).

Third and finally, the low-degree partial covering algorithm generally had a consistent, small advantage over the naive greedy baseline. Naturally, they performed essentially identically at the lowest and highest coverage rates as one would expect—at the lowest target coverage, both generally chose the best single term, and there is only one error rate for covering 100% of the data. Outside these extremes, recall that both algorithms were given the same target fractions: for the points at each threshold for the low-degree algorithm, the points for the corresponding thresholds for the baseline algorithm generally covered a larger fraction than necessary (shifted to the right) and suffered slightly greater error rates (shifted up). This matches our intuition that the low-degree algorithm works by discounting the points that are shared by many terms (that are likely to be chosen). Again, we stress that the baseline method also does not feature the same approximation guarantee as the low-degree algorithm.

Chapter 5

Apply the Algorithm on Real World Application: Anomaly Explanation Using Meta-data

We already introuced the anomaly explanation problem and it's data set in section 1.4. In this section, I am going to present the rustles for our real world data.

5.1 Algorithm Evaluation Procedure

We evaluated the aglorithm of producing conditions referring solely to the metadata that predict the image will be an anomaly.

To detect anomalies, we used principal component analysis(PCA). Following prior work [28], we defined anomaly scores by the reconstruction error using three principal components, and defined the top 3% scoring images as anomalies. We confirmed that there was no significant improvement to using five, ten, or twenty component nor lower percentage. For time intervals where the camera changed resolutions or moved, we obtained independent PCA deconstruction errors that would have detracted from the goal of finding a diverse set of metadata explanations. We evaluated our partial greedy set cover algorithm for this task. Meanwhile, we also evaluated three other algorithms for the task as baselines: the Patient Rule Induction Method (PRIM) [1], random forests [22], and tolerant elimination algorithm, which is a new algorithms from the artificial intelligence community that prioritize precision over recall, and thus was good candidates for this task. PRIM [12] is a classic statistical method to find a region of the input associated with high-valued values of a dependent output variable. We ran PRIM once with the real-valued metadata and PCA reconstruction errors and another time with real-valued metadata and binary representations of anomalies. PRIM has a peeling step to remove non-anomalous data and an optional pasting step to correct for over-peeling. Because PRIM failed to run within 24 hours in three attempts to include pasting, we omitted this step.

Random forests is a well-known, simple but effective classification method. Pevny and Kopp [30] use random forests to produce per-example explanations of anomalies, in contrast to our explanations that describe conditions that capture the variety of anomalies observed on a given camera. We simply used the standard random forest classifier as our baseline, which we anticipate to be more accurate but surely less interpretable than the actual method used by Pevny and Kopp. For this method we used binary representation of both metadata and anomalies.

Tolerant elimination algorithm, was introduced by Juba [17]. The algorithm seeks to find a k-DNF explanation for some fixed k, i.e., an AND of ORs of k "literals"—our Boolean attributes or their negations. It forms a working hypothesis that initially includes all possible terms of size k. It then tries to narrow this k-DNF down to the best definition by iteratively eliminating terms that have more false-positives than a bound calculated based on a user defined tolerance parameter and the predicted probability of hitting an outlier. The algorithm iteratively reduces the target coverage bound until it either finds a formula that approximately achieves the target coverage, or determines that the bound is too small for statistical validity. These tight bounds restrict the algorithm, leading it to report a very high error rate in comparison to the other algorithms used for evaluation on most datasets because it was not able to eliminate enough terms.

To recall that, our Red-blue partial cover algorithm finds k-DNF with high precision by ignoring terms that exceed a given false-positive threshold, ignoring examples that are false-positives for too many terms(using a corresponding threshold), and using a greedy selection of terms to classify a specified fraction of the points as positive. By ignoring the terms and points that may "share too much," the errors due to each term can be treated roughly as fixed costs for greedy procedure: by carefully choosing these thresholds, our new method has quadratically smaller error than tolerant elimination in the worst case. Moreover, we found that this algorithm substantially outperformed tolerant elimination on a synthetic benchmark.

5.2 Results

We then evaluated the four methods described in the previous section on the four data sets using a three-fold cross-validation. Thus, we only used 2/3 of the data sets for training, using the remaining 1/3 as a test set, and averaged the three results.

5.2.1 Semantic Explanations

We draw some examples of explanations from the pond location (camera #269). Here are two non-anomalous images from this camera.

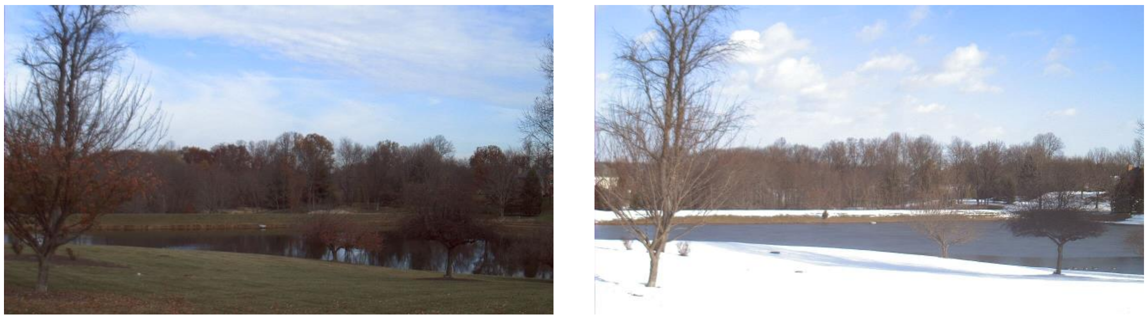
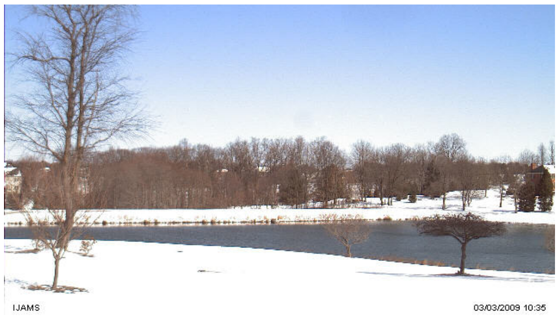
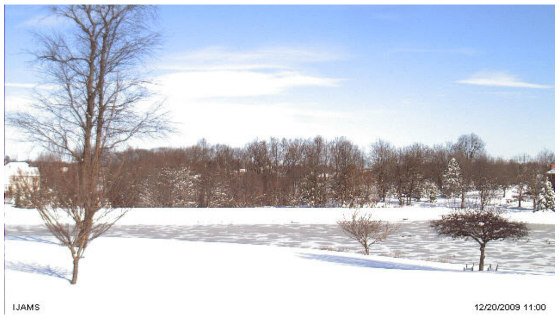


Figure 5.1: Non-anomalous images for camera #269

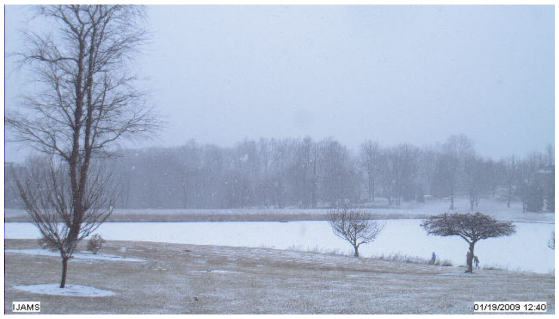
We first examine the terms generated as explanations of anomalies by our Red-blue partial set cover algorithm run using 2-DNF with 0.3% coverage; it obtained a precision error of 1.5% (i.e., out of the points satisfying the condition returned by the algorithm, only 1.5% were *not* anomalies). Since the data set only contains 3% anomalies, this condition captures essentially 1/10 of the anomalies. The four terms describing the condition, together with examples of images satisfying those individual terms, appear in Figure 5.2.



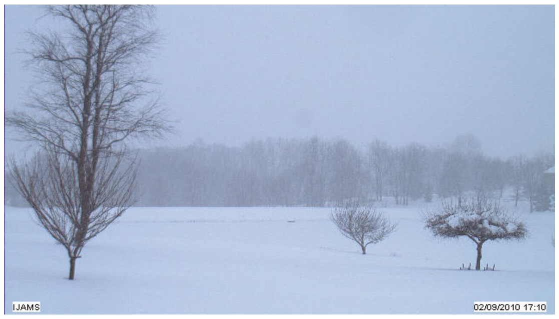
(a) **"panorama" ∧ "winter"**



(b) "community" ∧ "frost"



(c) "rain and snow" ∧ "holiday"



(d) "branch" ∧ "dry"

Figure 5.2: Terms covering 0.3% of images (1/10 of the anomalies) with 1.5% test precision error for camera #269, illustrated by example images. The terms contained in all of the three cross-validation runs are in bold.

With 0.2% coverage (1/15 of the anomalies), the test error rate was approximately 0%. This was obtained by the single term illustrated by the image in Figure 5.2a.

The 36 terms generated by our algorithm for 0.5% coverage contain the terms that appeared in 0.3% coverage, in Figure 5.2, together with the terms displayed in Figures 5.7 and 5.8. In this scenario, most of the scenes detected as anomalous were winter scenes or image encoding errors. As previously mentioned, the quality of explanations was negatively affected by image label quality. For example, for this camera, we see labels such as “farmhouse” and “ice boat,” and in the Toledo highway location, a truck was mistakenly identified as a “jet aircraft.”

We also examine the terms obtained for camera #623, the Moody Gardens theme park. A couple of ordinary images for this camera are shown below.

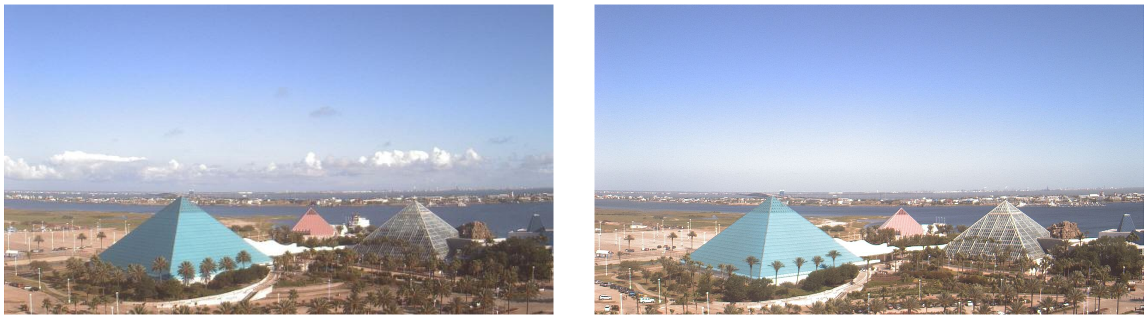
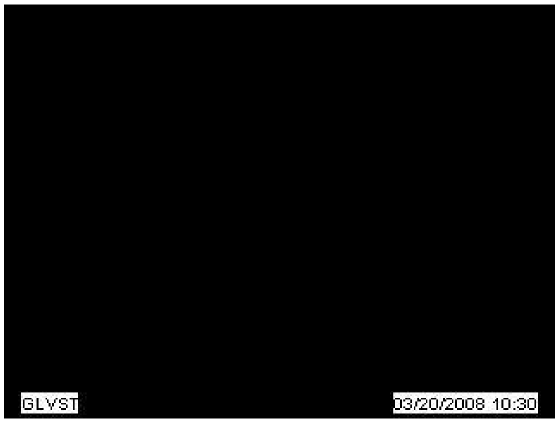
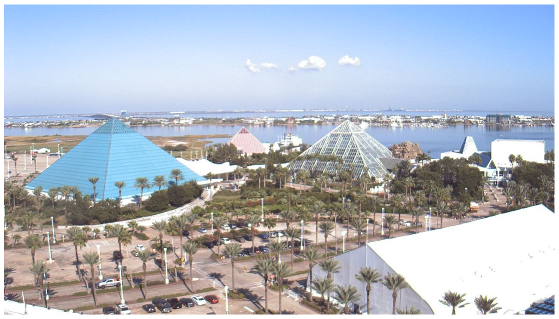


Figure 5.3: Non-anomalous images for camera #623

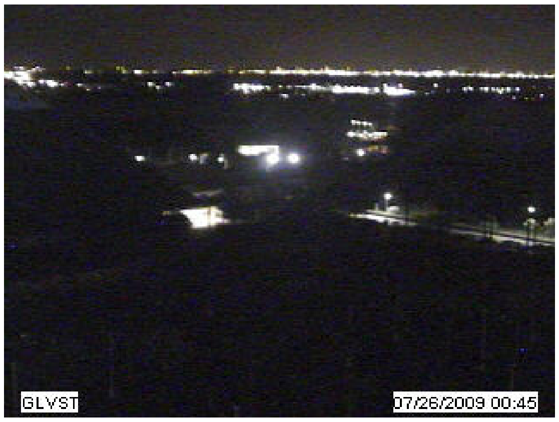
The 2-DNF explaining 0.1% of the anomalies for camera #623 is shown in Figure 5.4. Although we were able to obtain a similarly high rate of precision for covering 0.1% of the images (2.2% test error), as a consequence of the quality of the image labels, the anomalies are harder to interpret for this camera. The formulas covering 0.3%–0.5% of the images contained 84 and 98 terms, respectively, and are not included here.



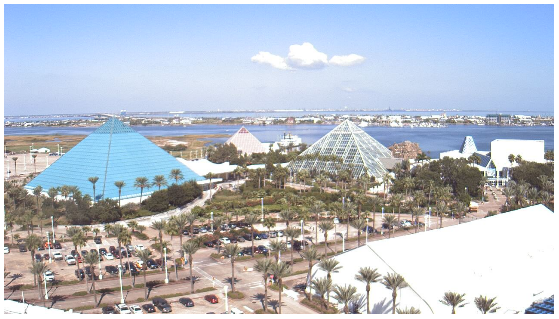
(a) **"text"** ∧ **"image"**



(b) **"bird's eye view"** ∧ **"peninsula"**



(c) "gadget" ∧ "weekend"



(d) **"horizon"** ∧ **"bird's eye view"**

Figure 5.4: Terms covering 0.1% of images (1/30 of the anomalies) with 2.2% test precision error for camera #623, illustrated by example images. The terms contained in all of the three cross-validation runs are in bold.

5.2.2 Quantitative Comparison of Methods

The performance of the methods on the four data sets are shown in Figure 5.5 and Figure 5.6. The performance of Juba's [17] tolerant elimination algorithm is not included in these graphs. It had an error rate ranging from 92% to 98% for all cameras and anomaly percentages, which was significantly worse than the other three methods.

Camera	Precision error with 1% coverage	Precision error with 3% coverage
#623	0.3930	0.7713
#269	0.1366	0.4082
#4312	0.7603	0.8521
#21656	0.8579	0.9110

Table 5.1: Precision error rates obtained by red-blue partial cover with 3% of elements defined as anomalies

Generally, for this task it appears that having a larger data set is extremely important for generating a set of explanations with low error rates. For instance, cameras #623 and #269 have 131873 and 123886 images respectively (Figure 5.5), whereas cameras #4312 and #21656 have 75453 and 73847 images respectively (Figure 5.6), and the latter have higher error rates as shown in the table below for our red-blue cover algorithm and in the precision error vs. coverage graphs. In any case we observe that for the larger data sets, it was possible to obtain sufficiently high precision to obtain high confidence that the rules we found really do indicate that an image will be an anomaly, as long as we aim to explain a moderately small fraction of the possible anomalies.

It also shows that random forest does much better when we are trying to fit a larger fraction of data set than our partial set cover algorithm. This is not surprising, since 2-DNF gives limited information. For our new algorithm, there are still some advantages over random forest. First, our new algorithm is competitive with random forest at smallish percentages. For example as camera 269, when the coverage percentage is very low, out algorithm yields a better result. The other interesting thing is that our algorithm has much easier interpretability than random forest. The chosen terms for our algorithm are easy to understand and interpret.

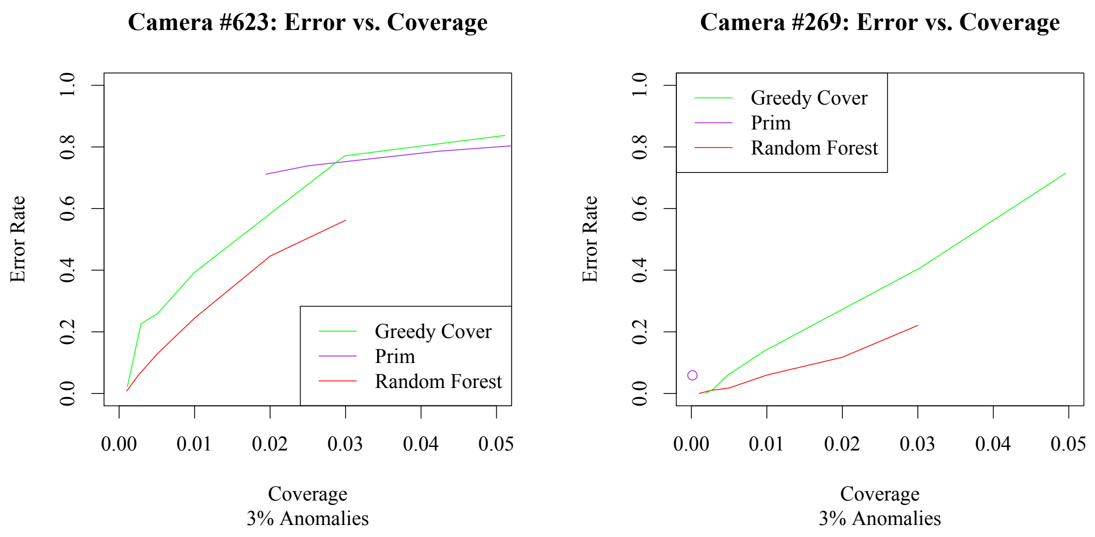


Figure 5.5: Precision error vs. coverage for larger datasets (>80k training examples). Observe that we can obtain greater than 75% precision as long as we only fit a fraction of the anomalies.

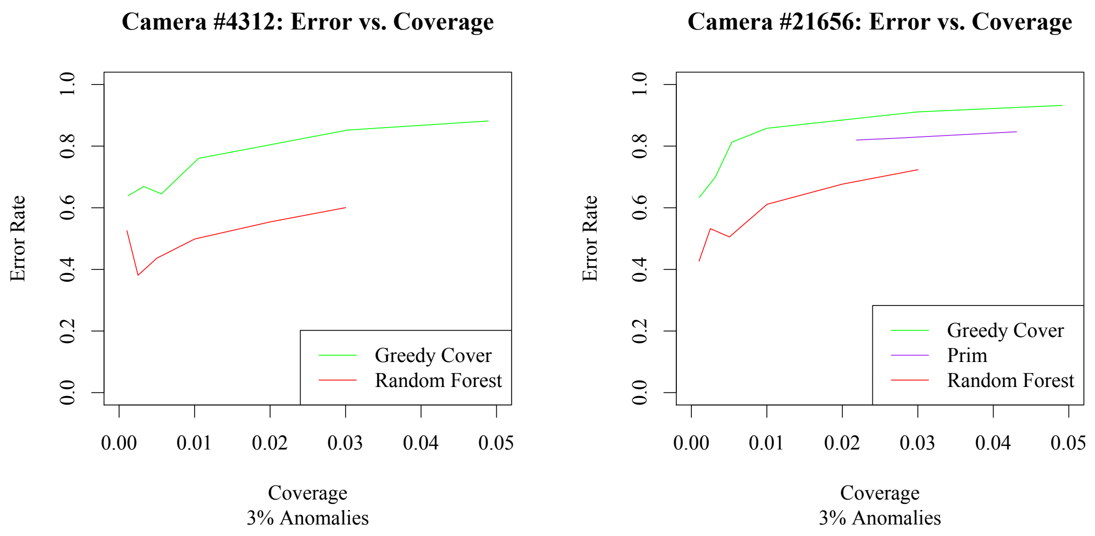


Figure 5.6: Precision error vs. coverage for smaller datasets ($\leq$50k training examples). Observe that no method obtains precision substantially greater than 60%.

(a) "phenomenon" $\wedge$ "texture"

(b) **"winter"** $\wedge$ **"woodland"**

(c) **"ice boat"** $\wedge$ **"frost"**

(d) "pasture" $\wedge$ "snow"

(e) **"shape"** $\wedge$ **"area"**

(f) **"community"** $\wedge$ **"winter storm"**

(g) "sky" $\wedge$ "winter storm"

(h) **"ice rink"** $\wedge$ **"estate"**

(i) "lavender" $\wedge$ "purple"

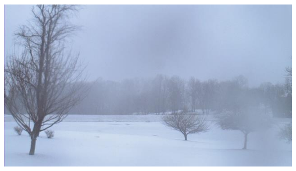
(j) "prairie" $\wedge$ "blizzard"

(k) "white" $\wedge$ "no precip"

(l) **"piste"** $\wedge$ **"no precip"**

(m) **"community"** $\wedge$ **"ice boat"**

(n) "farmhouse" $\wedge$ "winter"


(o) "event" $\wedge$ "fog"

(p) "weather" $\wedge$ "macro photography"

Figure 5.7: Example images and terms covering 0.5% of images (1/6 of anomalies) with 5.8% test precision error for camera #269. Terms that appeared in all three cross-validation runs appear in bold. *(Continued in Figure 5.8.)*

(a) "season ∧ "atmosphere of earth""

(b) **"cottage" ∧ "winter"**

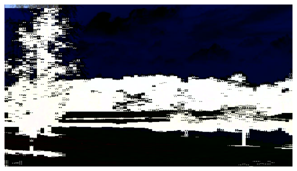
(c) **"yellow" ∧ "not cloudy"**

(d) **"ice boat" ∧ "sunset-night"**

(e) **"ice boat" ∧ "freezing"**

(f) "phenomenon" ∧ "watercolor paint"

(g) "yellow" ∧ "light"


(h) "font" ∧ "hot"

(i) "sky" ∧ "flock"

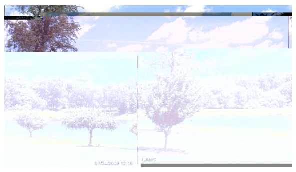
(j) "blizzard" ∧ "barn"

(k) **"prairie" ∧ "solarNoon-goldenHour"**

(l) "texture" ∧ "fog"

(m) "purple" ∧ "atmospheric phenomenon"

(n) "ice" ∧ "natural environment"


(o) **"horizon" ∧ "panoroma"**

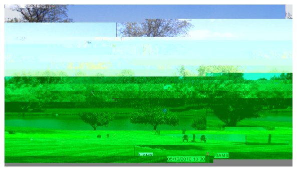

(p) "horizon" ∧ "solarNoon-goldenHour"

Figure 5.8: Example images and terms covering 0.5% of images (1/6 of anomalies) with 5.8% test precision error for camera #269, continued from Figure 5.7. Terms that appeared in all three cross-validation runs appear in bold.

Chapter 6

Conclusion and Further Work

6.1 For the Algorithm

We have exhibited an algorithm for the exception-tolerant variant of the learning abductive reasoning task introduced by Juba [17]. Our new algorithm: Red-Blue Partial Set Cover algorithm both achieves a substantially better error guarantee and performs substantially better on some challenging synthetic data tasks. It also performs well on some real world data set for Anomaly explanation of images. A natural question is how much scope remains to improve algorithms for this task. This question is wide open.

As a point of comparison, consider the standard agnostic supervised learning task in which our objective is merely to minimize classification errors. The best known algorithm for agnostic learning of k-DNF, due to Awasthi, Blum, and Sheffet [4] can achieve an approximation ratio of $n^{k/3+o(1)}$. By contrast, we only know that agnostic learning of k-DNF with *additive error* is intractable (an approximation *ratio* of ≈ 1). Even for agnostic learning of the much richer class of halfspaces, we only know that the task is intractable up to a ratio of $2^{\log^{1-\lambda} n}$ for $\lambda > 0$, which is still sub-polynomial, that is, less than any $n^{1/r}$ [11].

Now, if we restrict the form of the hypothesis to a k-DNF, it is likely that we can say much more; by contrast, the above results hold for the *improper* variant of the problem in which we do not restrict the form of the returned hypothesis. Again, taking agnostic supervised learning of k-DNFs as a point of comparison, Feldman [13] was able to show that finding a 1-DNF that obtains a $2^{\sqrt{\log n}}$ approximation ratio is intractable. Even so, again, a gap remains between these sub-polynomial approximation ratios for which we believe that the problem is intractable, and for the polynomial approximation ratios for which we possess algorithms.

6.2 For the Anomaly Explanation Task

Our main contribution is that we propose a new task, anomaly explanation using metadata. The key feature of this task is that it is *cross-modal:* we are seeking an explanation of anomalies in one type of data, in this case anomalies in image data, using conditions derived from other types of data. This allows us to connect data with a strong, well-understood semantics such as weather data, the date and time, or image labels, to a pixel-level model of anomalies.

Our second contribution is that we demonstrate that this task can be solved for a standard notion of anomalies in webcam data [28] by using algorithms that prioritize precision over recall, as long as the data set is relatively large, for example, using a training set containing more than 80k images: given such a large training set, it was able to find simple conditions that explain between 1/10 and 1/6 of the anomalies with precision greater than 75%, and 1/30 of the anomalies with precision greater than 97%. Using smaller training sets of size closer to 50k, none of the methods we considered were able to obtain greater than 60% precision, and thus could only address this task weakly at best. We note that since our task is inherently one of learning about events that occur by definition at most 3% of the time, it is not surprising that they should require a large training set (and indeed, also a large test set).

There are also several things we can improve for obtain a better result for the anomaly explanation task. First, the data set we are using is still too small; We only have four cameras with size from $70kto140k$. Second, we want to get more reasonable attributes for images and filter out the weird attributes. For example, as figure 5.4 (c) shows a image with label ”gadget”. We find ”gadget” isn't relative that image. We believe, if we can carefully filter out those weird attributes, our results would make much more sense.

Bibliography

[1] *R package PRIMsrc: Bump Hunting by Patient Rule Induction Method for Survival, Regression and Classification*, 2015.

[2] Vladimir Agafonkin. Suncalc api, 2015.

[3] Alphabet. Google vision api, 2016.

[4] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Improved guarantees for agnostic learning of disjunctions. In *Proc. 23rd COLT*, pages 359–367, 2010.

[5] Sakshi Babbar. Detecting and describing non-trivial outliers using bayesian networks. In *Proc. Cognitive Computing and Information Processing (CCIP)*.

[6] Václav Bahut. Outlier detection and explanation. Bachelor's thesis, Faculty of Informatics, Masaryk University, 2015.

[7] Robert D. Carr, Srinivas Doddi, Goran Konjevod, and Madhav V. Marathe. On the red-blue set cover problem. In *Proc. 11th SODA*, pages 345–353, 2000.

[8] The Weather Channel.

[9] Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence*. Addison-Wesley, Reading, MA, 1985.

[10] P. Cox and T. Pietrzykowski. Causes for events: their computation and applications. In *Proc. 8th Int'l Conf. Automated Deduction*, pages 608–621, 1986.

[11] Amit Daniely. Complexity theoretic limitations on learning halfspaces. In *Proc. 48th STOC*, pages 105–117, 2016.

[12] J-E. Dazard and J.S. Rao. Local sparse bump hunting. *J. Comp Graph. Statistics*, 19(4):900–929, 2010.

[13] Vitaly Feldman. Optimal hardness results for maximizing agreements with monomials. In *Proc. 21st CCC*, pages 226–236, 2006.

[14] J. Hobbs, M. Stickel, D. Appelt, and P. Martin. Interpretation as abduction. Technical Report 499, SRI, Menlo Park, CA, 1990.

[15] Nathan Jacobs, Nathaniel Roman, and Robert Pless. Consistent temporal variations in many outdoor scenes. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–6, 2007.

[16] Nathan Jacobs, Scott Satkin, Nathaniel Roman, Richard Speyer, and Robert Pless. Geolocating static cameras. In *Proc. International Conference on Computer Vision (ICCV)*, pages 1–6, 2007.

[17] Brendan Juba. Learning abductive reasoning using random examples. In *Proc. 30th AAAI*, pages 999–1007, 2016.

[18] Roni Khardon and Dan Roth. Learning to reason. *J. ACM*, 44(5):697–725, 1997.

[19] Edwin M. Knorr and Raymond T. Ng. Finding intensional knowledge of distance-based outliers. In *Proc. Very Large Data Bases Conference (VLDB)*, pages 211–222, 1999.

[20] Chia-Tung Kuo and Ian Davidson. A framework for outlier description using constraint programming. In *Proc. AAAI Conference on Artificial Intelligence*, pages 1237–1243, 2016.

[21] Douglas B. Lenat. CYC: a large-scale investment in knowledge infrastructure. *CACM*, 38(11):33–38, 1995.

[22] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[23] John McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1–2):27–39, 1980. Available at http://www-formal.stanford.edu/jmc/circumscription.html.

[24] Barbora Micenková, Xuan-Hong Dang, Ira Assent, and Raymond T. Ng. Explaning outliers by subspace separability. In *Proc. International Conference on Data Mining*, 2013.

[25] Pauli Miettinen. On the positive-negative partial set cover problem. *Information Processing Letters*, 108(4):219–221, 2008.

[26] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, New York, NY, 2014.

[27] Ryan O'Donnell and Karl Wimmer. Approximation by DNF: Examples and counterexamples. In *Automata, Languages, and Programming: 34th ICALP*, volume 4596 of *LNCS*, pages 195–206. Springer, Berlin, 2007.

[28] Joseph O'Sullivan, Abby Stylianou, and Robert Pless. Democratizing the visualization of 500 million webcam images. In *Applied Imagery Pattern Recognition Workshop (AIPR)*, 2014.

[29] David Peleg. Approximation algorithms for the label-covermax and red-blue set cover problems. *J. Discrete Algorithms*, 5:55–64, 2007.

[30] Tomáš Pevny and Martin Kopp. Explaining anomalies with sapling random forests. *Information Technologies - Applications and Theory Workshops, Posters, and Tutorials*, 2014.

[31] David Poole. A methodology for using a default and abductive reasoning system. *Int'l J. Intelligent Sys.*, 5:521–548, 1990.

[32] ryanss. Holidays library.

[33] Petr Slavík. Improved performance of the greedy cover algorithm for partial cover. *Information Processing Letters*, 64(5):251–254, 1997.

[34] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 18(11):1134–1142, 1984.

[35] Leslie G. Valiant. A neuroidal architecture for cognitive computation. *J. ACM*, 47(5):854–882, 2000.

[36] Leslie G. Valiant. Robust logics. *Artificial Intelligence*, 117:231–253, 2000.

Vita

Mengxue Zhang

Degrees	B.S. Virginia Tech, Computer Science, May 2015 M.S. Washington University St.Louis, Computer Science, May 2017
Publications	Mengxue Zhang, Tushar Mathew, Brendan Juba. *"An Improved Algorithm for Learning to Perform Exception-Tolerant Abduction"* **AAAI** (2017)

May 2017