ABSTRACT

Title of dissertation: COST-SENSITIVE INFORMATION ACQUISITION IN STRUCTURED DOMAINS

Mustafa Bilgic, Doctor of Philosophy, 2010

Dissertation directed by: Professor Lise Getoor
Department of Computer Science

Many real-world prediction tasks require collecting information about the domain entities to achieve better predictive performance. Collecting the additional information is often a costly process (money, time, risk, etc.) that involves acquiring the features describing the entities and annotating the entities with target concepts and labels. For example, document collections need to be manually annotated for document classification and lab tests need to be ordered for medical diagnosis. Annotating the whole document collection and ordering all possible lab tests might be infeasible due to limited resources or may prove unnecessary. Thus, we need to be selective about which entity we annotate and which features we acquire. In this thesis, I explore effective and efficient ways of choosing the right information to acquire under limited resources. Specifically, I develop and empirically evaluate algorithms for feature and label acquisition in structured domains.

For the problem of **feature acquisition**, we are given entities with missing features and the task is to classify them with minimum misclassification cost. The

likelihood of misclassification can be reduced by acquiring features but acquiring features incurs costs as well. The objective is to acquire the right set of features that balance acquisition cost and misclassification cost. Because finding the optimal solution is intractable in general, most previous approaches have been greedy. However, greedy approaches often get stuck in local minima and cannot naturally address the practical scenario where more than one feature needs to be acquired. I introduce a technique that can reduce the space of possible sets of features to consider for acquisition by exploiting the conditional independence properties in the underlying probability distribution.

For the problem of **label acquisition**, I consider two real-world scenarios. In the first one, we are given a previously trained model and a budget determining how many labels we can acquire, and the objective is to determine the right set of labels to acquire so that the accuracy on the remaining ones is maximized. In this setup, the entities appear in a network and acquiring the label of an entity helps us determine the correct labels of the other entities in the network. I describe a system that can automatically learn and predict on which entities the underlying classifier is likely to make mistakes and it suggests acquiring the labels of the entities that lie in a high density potentially-misclassified region. In the second scenario, we are given a network of entities that are unlabeled and our objective is to learn a classification model that will have the least future expected error by acquiring minimum number of labels. I describe an active learning technique that can exploit the relationships in the network both to select informative entities to label and to learn a collective classifier that utilizes the label correlations in the network.

COST-SENSITIVE INFORMATION ACQUISITION IN STRUCTURED DOMAINS

by

Mustafa Bilgic

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2010

Advisory Committee:
Professor Lise Getoor, Chair/Advisor
Professor Min Wu, Dean's Representative
Professor Hal Daumé III
Professor David Jacobs
Professor Dana Nau

Acknowledgments

First and foremost I thank my advisor, Lise Getoor, for her guidance on doing interesting research, writing good papers, presenting to a wide audience, finding a job, networking, and many other academic and non-academic issues. She has always been a very good mentor and tutor and I cannot express enough how much I learned from her.

I thank the LINQS members – Rezarta Islamaj, Indrajit Bhattacharya, Prihtviraj Sen, Louis Licamele, Galileo Mark Namata, Elena Zheleva, Hossam Shara, Walaa Moustafa, and Lilyana Mihalkova, for their mentorship, for very useful discussions, for the wonderful work atmosphere, and for the invaluable friendship.

I thank all my collaborators, especially David Jacobs, Paul Bennett, Ben Shneiderman, Lilyana Mihalkova, Louis Licamele, Prithiviraj Sen, Galileo Namata, and Daozheng Chen, for all the useful and fruitful discussions and insights. I learned a lot from them and enjoyed working with them.

I sincerely thank my wife, Ayse, for always being there for me, for her patience and support when I worked on weekends and late nights, for uplifting my spirit when experiments and ideas failed, and sharing my joy and happiness when things worked, and for her smiles that made my days. I also thank our little daughter, Zeynep, for being a powerful source of happiness and stress-reliever.

I thank those who cannot be thanked enough: my parents, Mehmet Bilgic and Ummugulsum Bilgic, for their never ending encouragement, support, and love.

Table of Contents

1 Introduction 1
- 1.1 Feature Acquisition During Inference for Non-Relational Data 5
- 1.2 Label Acquisition During Learning for Relational Data 7
- 1.3 Label Acquisition During Inference for Relational Data 8
- 1.4 Outline and Contributions of this Dissertation 9
  - 1.4.1 Feature Acquisition During Inference: Value of Information Lattice (`VOILA`) 10
  - 1.4.2 Label Acquisition During Inference: Reflect and Correct (`RAC`) 11
  - 1.4.3 Label Acquisition During Learning: Active Learning for Networked Data (`ALFNET`) 12

2 Related Work 14
- 2.1 Feature Acquisition 14
- 2.2 Label Acquisition 18
- 2.3 Other Related Areas 23

3 Feature Acquisition During Inference 25
- 3.1 Introduction 25
- 3.2 Notation and Problem Formulation 28
- 3.3 Value of Information Lattice (`VOILA`) 34
  - 3.3.1 Reducing the Space of Possible Sets 35
  - 3.3.2 EVI Computation Sharing 37
    - 3.3.2.1 Subset Relationships 38
    - 3.3.2.2 Information Pathways at the Underlying Bayesian Network 39
    - 3.3.2.3 Incremental Inference 41
  - 3.3.3 Constructing `VOILA` 41
    - 3.3.3.1 Construction Algorithm 43
    - 3.3.3.2 Analysis of `VOILA` Construction Algorithm 44
  - 3.3.4 Using `VOILA` for Feature-value Acquisition 46
- 3.4 Experiments 47
  - 3.4.1 Search Space Reduction 47
  - 3.4.2 Expected Total Cost Comparisons 48
- 3.5 Conclusion 57

4 Label Acquisition During Inference For Relational Data 59
- 4.1 Introduction 59
- 4.2 Problem Formulation 63
  - 4.2.1 Collective Classification 64
    - 4.2.1.1 Iterative Classification Algorithm (`ICA`) 66
    - 4.2.1.2 Pairwise Markov Random Fields (`MRF`) 67
  - 4.2.2 Label Acquisition 69

4.3 Active Inference . . . 71
4.3.1 Approximate Inference and Greedy Acquisition (AIGA) . . . 72
4.3.2 Viral Marketing Acquisition (VMA) . . . 74
4.3.3 Reflect and Correct (RAC) . . . 77
4.3.4 Generalized Utility-based Active Inference . . . 84
4.4 Experimental Evaluation . . . 86
4.4.1 Understanding Flooding . . . 86
4.4.1.1 Synthetic Data Generation . . . 88
4.4.1.2 The Spectrum of Flooding . . . 89
4.4.2 Experiments Comparing Different Active Inference Techniques 92
4.4.2.1 Experiments on Synthetic Networks with 200 Nodes 94
4.4.2.2 Experiments on Synthetic Networks with 2000 Nodes 95
4.4.2.3 Experiments on Real-world Datasets . . . 101
4.5 Summary and Contributions . . . 104
4.6 Conclusions . . . 106

5 Label Acquisition During Learning for Relational Data 108
5.1 Introduction . . . 108
5.2 Background . . . 111
5.2.1 Collective Classification . . . 111
5.2.2 Active Learning . . . 114
5.3 ALFNET . . . 114
5.4 Semi-supervision and Dimensionality Reduction . . . 119
5.5 Experiments . . . 120
5.5.1 Data . . . 120
5.5.2 Methodology . . . 121
5.5.3 Results . . . 123
5.5.3.1 Semi-supervision and Dimensionality Reduction . . . 123
5.5.3.2 Active Learning . . . 125
5.5.3.3 Ablation Studies . . . 127
5.5.3.4 Disagreement Computation Strategies . . . 129
5.6 Conclusion . . . 131

6 Conclusions and Future Work 133
6.1 Summary of Contributions . . . 133
6.2 Future Directions . . . 135
6.2.1 Complex Cost Structure . . . 135
6.2.2 Cost-sensitive Large-scale Data Mining . . . 136
6.2.3 Visualization Support . . . 137
6.2.4 Other Prediction Tasks . . . 139
6.3 Conclusion . . . 139

Chapter 1

Introduction

We often need to make decisions and take appropriate actions in a complex and uncertain world. Doctors need to diagnose illnesses for patients, judges need to decide whether the suspects are innocent or guilty, banks need to approve or deny loan applications, cell phones and computers need to recognize speech, and email clients need to detect spam. In all these examples and more, we strive to make accurate judgments and draw the conclusions by gathering as much useful information as possible; doctors order lab tests for their patients, judges analyze surveillance videos of the crime scenes, banks check the applicants' credit scores and income, and electronic devices and email clients are provided examples such as transcribed speech and email messages with spam flags.

Unfortunately, however, we are generally limited on the type and amount of information we can gather. Acquiring the necessary information might cost money, we might need to act quickly and thus cannot wait to gather and process all the information we like, and sometimes the information we seek might not be available; lab tests and credit checks cost money while watching a surveillance video, transcribing speech, and tagging emails with spam flags takes time and effort. Due to these constraints and limitations, we need to find the right balance between making the optimal judgments and acquiring more information. This requires us to be se-

lective about which information we should gather and process so that the acquired information will let us reach the best conclusion possible; for e.g., performing all possible lab tests and watching the whole surveillance video might not be possible, cost-effective, or even necessary.

The artificial intelligence field in general, and the machine learning, decision making under uncertainty, and planning subfields in particular provide useful machinery for drawing the right conclusions and information acquisition decisions. In this thesis, I focus on a simple yet quite common type of judgment called *classification*. In classification, a situation or an entity, which I generically refer to as the instance, is described by a set of *features* and the task is to choose one of pre-specified categorical alternatives as the judgment, which is called the *label*. For example, a patient is described by his/her demographic information as well as lab test results, and one of a set of pre-specified sicknesses is chosen as the patient's label. Using this terminology, the judgment that we make about each instance (patient, suspect, speech, email, etc.) is to predict its label, and we gather as much useful information (features and labels) as possible to choose the best label.

To formally define and determine the right balance between a "good" classification and the right choice and amount of information to gather, we need to determine when a classification is considered "good." In theory, a classification is obviously good if it is correct. Thus, conversely, we pay a penalty when the classification is incorrect, and this penalty is called the *misclassification cost*. However, whether a classification is incorrect is typically not known in practice; thus, a probabilistic version, *expected misclassification cost*, is used instead. Given the right

modeling assumptions, the more information is acquired, the less probable a misclassification gets. This inverse relationship calls for determining the right trade-off between higher acquisition costs and lower misclassification costs, a task I refer to as *cost-sensitive information acquisition for classification*.

Cost-sensitive information acquisition can be setup in a few different ways depending on the application and domain. In some cases, we are given both the information costs and misclassification costs and we would like to keep acquiring information as long as the expected reduction in misclassification cost outweighs the cost of the information. In other cases, we are given a budget to spend and we need to acquire information so as not to exceed that budget. In other cases, we are given a target classification performance, and we need to acquire as much information as needed to reach that target performance. The choice of the setup is often task and application dependent.

Classification typically has two phases, the *learning phase* and the *inference phase*, and what information can be acquired depends on what phase we are in. In the *learning phase*, the classifier is presented with example instances whose feature values and labels are known. The classifier is learned as a mapping from the feature values to the labels. Examples include patients with their lab test results and the correct diagnosis, recorded speech with its transcription, and email text with the correct spam flag. In this phase, we need to acquire both features and labels to be able to learn the correct classification function for the underlying domain.

The second phase is called the classification or the *inference phase*. In this phase, the classification function, which has been trained in the learning phase, is

presented with just the feature values and it is expected to determine the correct labels. In this phase, we definitely want to acquire (some of) the features. Depending on the domain and application, however, we might also want to acquire the labels of a few instances. For a *non-relational domain* where each instance is treated to be independent of the other instances during inference, the labels of the instances that are most likely to be predicted incorrectly can be acquired if the classification is part of a critical decision-making application. In *relational domains*, on the other hand, the instances and their labels are related to one another and thus the label of an instance can be used as a feature for a related instance. For example, recognizing a word in a given speech segment can benefit from recognizing other words in the segment; in this case, acquiring the labels of a subset of the words can help for correct recognition of the remaining ones.

When we take the cross product of different scenarios, i.e., {feature acquisition, label acquisition}, {during inference, during learning}, and {for non-relational data, for relational data}, we have eight possibilities. In this thesis, I focus on three scenarios that I think are the most practical and yet relatively under-researched. These are, feature acquisition during inference for non-relational data, label acquisition during learning for relational data, and label acquisition during inference for relational data. Next, I describe these scenarios in more detail.

1.1 Feature Acquisition During Inference for Non-Relational Data

In this scenario, we assume we have an already trained classifier and we are given instances with missing feature values for which we need to determine the correct labels. We are also given the cost functions for feature acquisitions and misclassifications. Our task is to devise a policy that specifies which features to acquire in what order for a given instance and when to stop acquiring information and determine a label for it. We assume the instances appear in a non-relational setting, i.e., the acquisition and classification decisions can be made independently for each instance. A prototypical example of this problem is medical diagnosis, where a patient visits a doctor and the doctor needs to order lab tests to determine the correct diagnosis for the patient; the doctor needs to be selective about which lab tests to order for the patient.

The feature acquisition costs can have various levels of complexity, as discussed in detail by Turney (2000). In the simplest case, each feature has an independent cost and the cost of a set of features is just an additive cost. In a more realistic setup, the features can belong to groups and the first feature to acquire from its group incurs an overhead cost. For example, features regarding to various blood measurements can belong to the blood group and the first feature to acquire from that group incurs the overhead cost of drawing blood from the patient. Similar group structure is typical in other domains as well, such as car repair, where the first of the features that require dismantling the engine incurs an overhead cost; once the engine is dismantled, then it is cheaper to make various measurements. More

complex cost structures are also possible, such as the cost of acquiring a feature depending on the specific values of the previously acquired features. For example, a treadmill test measuring the stress level of the patient can be risky for patients of certain age.

As the last example suggests, the costs do not have to be just monetary costs. Sometimes, the costs can represent risk. In other cases, the patients can have preferences for or against certain procedures, and the cost in this case represents inability to accommodate the preference. Finally, the cost can be the wait time for the lab test results. Because it is nontrivial address all these different types of costs simultaneously, existing feature acquisition research typically focuses on only monetary costs.

Under relatively general assumptions, determining the optimal order in which features should be acquired and when to stop acquisition is intractable. Features typically interact with the class label in complex ways; the benefit that a set of features provides can be more than just the sum of the benefits of the individual features; a single feature might not be useful, but a set of features can be very useful when ordered together. Similarly, the cost structure can complicate devising the optimal policy; the cost of a feature can change based on what other features are acquired and what the values of the features turn out to be. Thus, finding the optimal solution typically requires considering all subsets of features as well as all possible orderings of all possible outcomes. Because of these intractable computations, most of the previous approaches to feature acquisition have been greedy, considering each

feature in isolation (Gaag and Wessels, 1993). More discussion of the related work on feature-value acquisition is provided in Chapter 2.

1.2 Label Acquisition During Learning for Relational Data

For many domains, the feature descriptions are freely available, however, the class label of the instances requires costly processing. For example, it is generally easy to extract documents from the web, speech data can be collected easily, and email is abundant. But, annotating the emails with the correct spam tag and transcribing speech is a tedious process that takes time and effort. This is true for label acquisition during both learning and inference.

The objective for label acquisition in the learning phase is to reduce the label acquisition cost while learning the correct classification function. The problem is typically cast as, given a budget determining the number of labels that can be acquired, determine the right set of labels to acquire so that we learn the best possible classifier. This problem is called "active learning" in contrast to passive learning where the training data is assumed to be given. There is ample work on active learning for cases where the instances are assumed to be independent and identically distributed (i.i.d.). A few examples include uncertainty sampling (Lewis and Gale, 1994) where the label of the instance on which the existing model is most uncertain is acquired, query by committee (Seung et al., 1992) where the label of the instance on which a committee of classifiers disagrees the most is acquired, and estimated error reduction (Roy and McCallum, 2001) where the label of the instance

that is expected to provide the most reduction in future error (i.e., misclassification) is acquired. A more comprehensive discussion of the related work on active learning is provided in Chapter 2.

However, instances often participate in complex relationships in real-world settings. For example, words in a sentence form a sequence, suspects have connections, and the loan applicants have co-workers. Classification systems that exploit the relationships between instances have recently been explored under the name "collective classification" (see Sen et al. (2008) for an overview) and it has been shown that the classification performance can be significantly improved by using relational information. Active learning that takes the relationships between instances into account is comparatively under-researched. Even though there has been some work done on active learning for sequence data (Settles and Craven, 2008), active learning techniques that fully exploit the arbitrary structure in relational data are still needed. In this thesis, I provide a general purpose active learning algorithm that utilizes the relationships between the instances for both classification and determining which instances' labels to query.

1.3 Label Acquisition During Inference for Relational Data

Label acquisition has been largely studied for the learning phase for historical reasons. Most of the previous learning techniques assumed i.i.d. data; hence, acquiring the label of an instance in the i.i.d. setting in the *inference phase* is not helpful for the remaining instances. However, with the recent surge of network data and the

collective models that exploit the relationships between instances, the classification decisions about related instances are now interdependent. Thus, acquiring the label of an instance at inference time can be very helpful for the remaining instances. For example, providing the label of word in a sequence can be very helpful for the labels of the related words. Acquiring the label of an instance at inference time is referred as "active inference" (Rattigan et al., 2007b) in contrast to "active learning."

Finding the optimal set of labels to acquire has been proven to be intractable even for polytrees by Krause and Guestrin (2005a). The complexity of finding the optimal solution is expected to be harder for networks of arbitrary structure. Rattigan et al. (2007b) propose to cluster the network into k clusters and acquire the labels of the centroids of the clusters. In this thesis, I develop an acquisition technique that can learn and predict when the underlying collective model is likely to make mistakes and acquires the label of a central instance in a misclassified region of the network.

1.4 Outline and Contributions of this Dissertation

In this thesis, I develop cost-sensitive information acquisition techniques for i) feature acquisition during inference, ii) label acquisition during inference (active inference), and iii) label acquisition during learning (active learning). I developed three systems and algorithms to tackle these problems. I listed these techniques in Table 1.1.

Table 1.1: The contributions of this thesis.

	Feature-value Acquisition	Label Acquisition
During Learning	–	ALFNET
During Inference	VOILA	RAC

For the feature-value acquisition problem, I developed a data structure called Value of Information Lattice (VOILA) and the associated algorithms with it, for the active inference problem, I developed Reflect and Correct (RAC), and for the active learning problem, I developed Active Learning for Networked Data (ALFNET). Next, I briefly describe these techniques.

1.4.1 Feature Acquisition During Inference: Value of Information Lattice (VOILA)

In this setup, we are given an instance to classify and (some of) the feature values describing the instance are missing but can be acquired at a cost. The objective is to develop an acquisition policy that minimizes the sum of feature acquisition cost and misclassification cost. The optimal solution requires us to consider all possible orderings of all possible values of all possible features, which makes finding the optimal solution intractable in general. Many of the previous approaches thus have been greedy, considering one feature at a time. However, due to its shortsightedness, greedy cannot estimate the usefulness of combinations of futures and can prematurely reject acquiring more information.

In Chapter 3, I present a technique that makes reasoning with sets of features tractable in practice. The idea is based on the intuition tat certain features render other features useless in real-life settings. For example, ordering a more expensive lab test might render the information that can be obtained from a cheaper one useless. Similarly, certain lab tests might be useful only if they are ordered in combination with other lab tests. We infer these relationships from the data by learning a Bayesian network and posing independence queries to the underlying probability distribution (Bilgic and Getoor, 2007). The approach that we propose, which we call Value Of Information Lattice (`VOILA`), significantly reduced the search space, made reasoning with sets of features possible, and outperformed the greedy approaches on several real-world medical datasets. I discuss `VOILA` in detail in Chapter 3.

1.4.2 Label Acquisition During Inference: Reflect and Correct (`RAC`)

In the problem of label acquisition during inference, we are given a learned model of the domain and a budget to spend to acquire labels for some of the instances. Our objective is to acquire the labels of the right set of instances so that in the end, the performance on the remaining ones is maximized (or equivalently, the misclassification cost on the remaining ones is minimized).

I developed a technique called Reflect and Correct (`RAC`) that can learn and predict when the underlying model is likely to make mistakes (Bilgic and Getoor,

2008, 2009, 2010). Rather than using the probability estimates of the underlying model directly as an estimate of misclassification, I treat it as one of the few features that indicate a possible misclassification. I also construct a few other features that are possible indicators of misclassification and use these features to learn a classifier that can predict whether a given instance is misclassified. `RAC` then suggests acquiring the labels in the region where the most number of misclassifications are made are. I empirically show that `RAC` significantly outperforms several competitive baselines on both synthetic and real datasets. The `RAC` method is explained in detail in Chapter 4.

1.4.3 Label Acquisition During Learning: Active Learning for Networked Data (`ALFNET`)

In the problem of label acquisition during learning, we are given a set of instances whose labels are missing and a budget that determines how many instances we can label. The objective is to acquire the labels for the right set of instances so that in the end we can learn the best possible model under the given budget. Most previous methods dealt with i.i.d. data. I developed an active learning technique for networked data (`ALFNET`) that can take the relationships in a network into account for both determining which labels to acquire and classifying related instances (Bilgic et al., 2010).

`ALFNET` works as follows. It makes sure that the acquired labels will be a representative sample of the underlying domain by clustering the network into a

small number of groups and distributing the label acquisitions across clusters. To determine which instances to label in each cluster, `ALFNET` capitalizes on the disagreement between a classifier that uses the relationships in the network, another classifier that uses only the local information about each instance, and the majority label in the cluster. I describe `ALFNET` in more detail in Chapter 5.

The rest of the thesis is organized as follows: I discuss related work on feature acquisition and label acquisition in Chapter 2. Then, I discuss our techniques for feature acquisition during inference, label acquisition during inference, and label acquisition during learning in Chapters 3, 4, and 5 respectively. Finally, I discuss future directions and conclude in Chapter 6.

Chapter 2

Related Work

In this chapter, I review related work on cost-sensitive feature and label acquisition. I first discuss feature acquisition and then move to label acquisition. Finally, I discuss other related areas that deal with missing and scarce information.

2.1 Feature Acquisition

In the feature acquisition problem, we are given instances with missing feature values and the task is to classify these instances. Misclassifying the instances is costly and the less we know about the instances, the more likely we are to misclassify them. We are given the option to acquire the values of the missing features, however, acquiring features is costly as well. The task is to devise an acquisition policy that can both decide which features to acquire in what order, and when to stop acquisition and classify the instance. This policy can be best thought of as a decision tree where each non-leaf node represents a feature acquisition, each edge represents an observed value of the acquired feature, and each leaf node represents a classification. The objective is devise the optimal policy that has the minimum combined cost of acquisition and misclassification.

Decision theoretic value of information calculations provide a principled methodology for information gathering in general (Howard, 1966; Lindley, 1956). Influence

diagrams, for example, are popular tools for representing decisions and utility functions (Howard and Matheson, 1984). However, because devising the optimal acquisition policy (i.e., constructing the optimal decision tree) is intractable in general, most of the approaches to feature acquisition has been myopic (Dittmer and Jensen, 1997), greedily acquiring one feature at a time. The greedy approaches typically differ in i) the problem setup they assume, ii) the way the features are scored, and iii) the classification model being learned. I review some of the existing work here, highlighting the differences between different techniques in these three dimensions.

Gaag and Wessels (1993) considers the problem of "evidence" gathering for diagnosis using a Bayesian Network. In their setup, they gather evidence (i.e., observe the values of the variables) until the hypothesis is confirmed or disconfirmed to a desired extent. They propose an acquisition algorithm that greedily computes the expected utility of acquiring a feature and chooses the one with the highest utility. They define the utility as the absolute value of the change in the probability distribution of the hypothesis being tested.

It is important to note that the distinction between a feature and a label is fuzzy in joint probabilistic models such as a Bayesian Network. We refer to a random variable as the label if it represents the class variable to be predicted and the random variables that help for predicting the label as the features, following the terminology used in classification. Given this terminology, in a Bayesian Network, the query variable(s) is the label and the rest are the features; feature acquisition problem is then deciding which feature variables to observe and include as evidence. Depending on the application, any of the variables can act as a label or a feature.

Núñez (1991) introduces a decision tree algorithm called EG2 that is sensitive to the feature costs. Rather than splitting the decision tree at a feature that has high information gain, EG2 chooses a feature that has least "information cost function," which is defined as the ratio of a feature's cost to its discriminative efficiency. EG2 is, however, is not directly optimized to balance the misclassification cost and feature acquisition cost; rather it is optimized for 0/1 loss while taking the feature costs into account. Similarly, Tan (1990) modifies the ID3 algorithm (Quinlan, 1986) to account for feature costs. Tan considers the domain where a robot needs to sense, recognize, and act, and the number of features is very large. For the robot to act efficiently, it needs to trade-off accuracy for efficiency.

Turney (1995) builds a decision tree called ICET (standing for Inexpensive Classification with Expensive Tests) using a genetic search algorithm (Grefenstette, 1986) and using Núñez (1991)'s criteria to build C4.5 decision trees (Quinlan, 1993). Unlike Núñez, Turney takes misclassification costs into account (in addition to the feature costs) to evaluate a given decision tree and looks for a good decision tree using genetic search algorithms.

Yang et al. (2006) build cost-sensitive decision trees and Naive Bayes classifiers that take both feature costs and misclassification costs into account. Unlike Núñez (1991), who scores features based on information gain and cost ratio, Yang et al. score features based on expected reduction in total cost (i.e., sum of the feature cost and misclassification cost) on the training data. By doing so, they take feature costs and misclassification costs into account directly at learning time.

Bayer-Zubek (2004) formulate the feature acquisition problem as a Markov Decision Process and provides both greedy and systematic search algorithms to develop diagnostic policies. Bayer-Zubek takes both feature cost and misclassification costs into account and automatically finds an acquisition plan that balances the two costs. She introduces an admissible heuristic for AO* search and describes regularization techniques to reduce overfitting to the training data.

Saar-Tsechansky et al. (2009) considers active feature acquisition for classifier induction. Specifically, they are given a training data with missing feature values, and a cost matrix that defines the cost of acquiring each feature value, they describe an incremental algorithm that can select the best feature to acquire iteratively to build a model that is expected to have high future performance. The utility of acquiring a feature is estimated in terms of expected performance improvement per unit cost. The two characteristics that make this work different from most of the previous work is that i) the authors do not assume a fixed budget apriori; rather they build the model incrementally, ii) each feature can have a different cost for each instance.

Finally, Greiner et al. (2002) analyze the sample complexity of dynamic programming algorithms that performs value iteration to search for the best diagnostic policies. They analyze the problem of learning the optimal policy, using a variant of the probably-approximately-correct (PAC) model. They show that the learning can be achieved efficiently when the active classifier is allowed to perform only (at most) a constant number of tests and show that learning the optimal policy is often intractable in more general environments.

2.2 Label Acquisition

For the label acquisition problem, we are typically given a budget to spend on acquiring labels and we need to determine which instances' labels should be queried in order to i) learn the best possible classification function (learning phase), ii) achieve the best performance on the remaining instances (inference phase) while operating within that budget. Label acquisition during learning is called "active learning" where the classifier is considered "active" in choosing the training data for itself, and it is called "active inference" if the labels are acquired during inference. I review some of the related work on active learning and active inference in this section.

Active learning is considered in three typical scenarios (Settles, 2009): i) membership query synthesis, ii) stream-based selective sampling, and iii) pool-based active learning. In the membership query synthesis setting, the active learner constructs a "synthetic" example instance with certain feature values and asks the oracle for the label of that instance (Angluin, 1988, 2001). In the stream-based selective sampling, the active learner draws an example instance from the actual data distribution and then the learner decides whether to ask the label for it or discard it (Cohn, 1996). In the pool-based active learning scenario, a large collection of unlabeled instances is available, which is also called the pool, and the active learner requests the label for a member of the pool (Lewis and Gale, 1994). In this chapter, I focus on related work that is most relevant to the approaches that I propose. Please see (Settles, 2009) for a comprehensive and running survey of active learning.

In all three scenarios, the active learner associates a score with the instance to label and queries the label of the instance with the best score. Techniques typically differ in how they score an instance. In most of the approaches, the instance that is expected to reduce the version space (Mitchell, 1982) the most is chosen to be labeled. However, because exact computation of the version space is nontrivial even for most simple classifiers, most techniques employ heuristic approaches to score the instances.

One of the early approaches is *query-by-committee*, probably first discussed in the active learning setting by Seung et al. (1992). In the query-by-committee approach, a committee of classifiers is maintained, and the instance on which the committee members disagree the most is chosen to be labeled. When the classifiers are chosen from the same hypothesis space, query-by-committee reduces the version space with each label acquired. Seung et al. (1992) use Gibbs sampling to sample hypothesis from the model distribution. Abe and Mamitsuka (1998) form the committee members using the bagging (Breiman, 1996) and boosting (Freund and Schapire, 1997) techniques. Similarly, Cohn et al. (1994) keep two hypothesis, the most general one and most specific one, and acquire a label on the region where these two hypotheses disagree. Finally, Tong and Koller (2001) approximate the version space for Support Vector Machines (Cortes and Vapnik, 1995) and acquire the label for the instance that is expected to reduce the version space the most.

Another important and arguably the most commonly used active learning strategy due to its simplicity is *uncertainty sampling* (Lewis and Gale, 1994). In this technique, the classifier is learned on the existing labeled data (a small number

of instances can be chosen at random initially), and then the label of the instance on which the existing classifier is most uncertain is acquired. For probabilistic classifiers, uncertainty can be measured using entropy (Shannon, 1948) or conditional error; for margin based approaches, such as Support Vector Machines, uncertainty can be measured as the distance from the separating hyper-planes (Tong and Koller, 2001).

In addition to query-by-committee and uncertainty sampling, many other important sampling techniques have also been proposed. Roy and McCallum (2001) score the instances based on how much they are expected to reduce the expected future error. Similarly, Settles et al. (2008) score instances based on how much they are expected to change the model parameters. Cohn et al. (1996) score the instances based on how much they are expected to reduce the variance (Geman et al., 1992) and Cohn (1997) scores the instances based on how much they are expected to reduce the bias.

An important obstacle for active learning is the possibility of acquiring the labels of outlier and noisy instances. This problem is especially present for techniques that are based on version space reduction, uncertainty sampling, query-by-committee, and expected model change. One way to tackle this problem is to weight the scores of each instance by their "representative-ness." For example, Settles and Craven (2008) weight the uncertainty score of an instance by its average similarity to the other instances in the pool. Methods that exploit the cluster structure in the data also address this problem (Nguyen and Smeulders, 2004; Dasgupta and Hsu, 2008).

Most of the approaches described above considered active learning for independent and identically distributed (IID) data. There has been growing interest on cases where the instances form a sequence such as text (Anderson and Moore, 2005; Baldridge and Osborne, 2004; Culotta and McCallum, 2005; Roth and Small, 2006; Scheffer et al., 2002; Settles and Craven, 2008; Settles et al., 2010). Anderson and Moore (2005) and Scheffer et al. (2002) discuss the framework and algorithms for active learning of Hidden Markov Models. Baldridge and Osborne (2004) address the problem that the data collected using an active learner for a model might not be ideal to learn another model. Culotta and McCallum (2005) consider the setup where annotating certain segments of text can be more difficult than others. Finally, Settles and Craven (2008) evaluate several active learning algorithms for probabilistic sequence models such as Conditional Random Fields (Lafferty et al., 2001).

Active learning for networks of arbitrary structure, however, is relatively underresearched. Zhu et al. (2003) extend the expected future error reduction technique of Roy and McCallum (2001) to graph data for Gaussian Random Fields. However, a practical implementation of this technique requires fast incremental training of the underlying classifier; Zhu et al. show how it can be done for Gaussian Random Fields, but it is unclear how to extend it to other relational classification models.

Another important setup for label acquisition, in addition to active learning, is *active inference*, where a few labels are acquired to help classifying the rest of the instances at inference time. As we have discussed in Chapter 1, active inference makes the most sense for data and models where the labels of instances are interdependent,

so that the acquired labels help for classifying the remaining instances during inference. Krause and Guestrin (2005a) discuss this problem in the context of value of information calculations in graphical models. They associate a reward for observing the value at a node, which is equivalent to acquiring a label, and they show that reward computations are #**P**-complete even for discrete polytrees. They also show that finding the optimal set to acquire in batch mode, where the acquisition decisions are made all at once, is $\mathbf{NP}^{\mathbf{PP}}$-complete for discrete polytrees. Finally, they show that finding the optimal set in a conditional plan, that is the next acquisition is conditioned on the previous acquisitions, is $\mathbf{NP}^{\mathbf{PP}}$-hard for discrete polytrees. For arbitrary networks, such as citation, friendship, and protein networks, finding the optimal solution is expected to be at least as hard as, if not harder than, considering discrete polytrees. In later work, Krause and Guestrin (2005b) show that the greedy acquisition provides a constant factor $(1 - \frac{1}{e-\epsilon})$ guarantee for submodular (Nemhauser et al., 1978) reward functions.

Rattigan et al. (2007a) tackle the active inference problem for networks of arbitrary structure and proposes a heuristic acquisition scheme, which first clusters the nodes of the network using a variant of k-means clustering adapted for network data, and acquires the labels of the centroids of the clusters. They show that the proposed method outperforms acquisition techniques based on network centrality measures such as degree, betweenness, and closeness centrality (Freeman, 1979).

2.3 Other Related Areas

Troubleshooting:

Troubleshooting address the problem of given a malfunction, find the causes and execute the necessary repair actions to fix the faulty system (Heckerman et al., 1995). It is closely related to the problem of feature acquisition. The main differences can be summarized as follows. In troubleshooting, the system state is given and the causes need to be found. In feature acquisition, however, the state (label) needs to be predicted as well as the features need to be acquired. Additionally, in addition to the pure feature acquisition actions, troubleshooting also has repair actions that can change the state/label of the system. In feature acquisition, acquiring a feature is typically assumed not to change the label of an instance.

Imputation:

Imputation is the substitution of missing feature values for instances (Rubin, 1987). Typical imputation techniques include substituting the mean, median, and mode of the value, which is computed from the observed feature values for other instances in the domain. It addresses the same problem that feature acquisition addresses, i.e., missing values; however, feature acquisition considers the possibility of acquiring the feature at a cost. Additionally, it is expected that imputation might not be the best solution for domains where feature acquisition is possible, such as medical domains.

Viral Marketing:

Viral marketing looks at the problem of given a social network of customers, which customers should be targeted for advertisement (or free samples) of a product so that the sales of the product in the network is maximized (Richardson and Domingos, 2002; Kempe et al., 2003; Leskovec et al., 2007a; Provost et al., 2007). Viral marketing is closely related to active inference in essence; viral marketing aims at maximizing the number of customers in the "buy" state, whereas active inference aims at maximizing the number of instances in the "correctly classified" state. However, there are fundamental differences between the two that warrant specific techqniues for both. For example, the instances in the viral marketing are typically either in a binary state or on a continuous scale, whereas in active inference the instance labels are categorical, and the number of categories can be an arbitrary number. Moreover, the dynamics of product adaptation in viral marketing versus label propagation in active inference are likely to be different.

Semi-supervised Learning:

Semi-supervised learning is the name of the technique that uses both labeled and unlabeled data for learning (Chapelle et al., 2006). The premise behind this technique is that labeled data is scarce and unlabeled data is abundant. Active learning also assumes the same setup that labeled data is scarce. Thus, techniques that combine active learning with semi-supervised learning are expected to be superior (Zhu et al., 2003).

Chapter 3

Feature Acquisition During Inference

In this chapter, I discuss feature-value acquisition for classification. Specifically, I assume that we are given a probabilistic model of the domain and an instance whose feature values are missing but can be acquired at a cost. I also assume that there is a misclassification cost in addition to the feature acquisition cost. I discuss how we can devise a feature acquisition and classification policy that determines which features to acquire for each instance and when to stop the acquisition and classify the instance.

3.1 Introduction

We often need to make decisions and take appropriate actions in a complex and uncertain world. An important subset of decisions can be formulated as a *classification* problem, where an instance is described by a set of features and one of finite categorical options is chosen based on these features. Examples include medical diagnosis where the patients are described by lab tests and a diagnosis is to be made about the disease state of the patient, and spam detection where an email is described by its content and the email client needs to decide whether or not the email is spam.

Much research has been done on how to learn effective and efficient classifiers assuming that the features describing the entities are fully given. Even though this complete data assumption might hold on a few domains, such as classifying an email, the features describing the entities might not be known initially for certain domains such as medical diagnosis, but the features can be acquired at cost. In addition to the feature costs, typically misclassifications have costs as well. In this setting, one is faced with devising a policy that describes which features should be acquired, in which order, and when to stop and classify the instance.

Devising the optimal policy in general requires considering all possible orderings and combinations of the features and their values. To provide some intuition, some features might be useful only if acquired together, and the cost of acquiring features can depend on which other features have been acquired. Because devising the optimal policy is intractable in general, previous work has been greedy (Gaag and Wessels, 1993; Yang et al., 2006), has approximated value of information calculations (Heckerman et al., 1993), and has developed heuristic feature scoring techniques (Núñez, 1991; Turney, 1995). Please see Section 2.1 for a more comprehensive overview of the related work.

The greedy approach, however, has at at least two problems. First, because it considers each feature in isolation, it cannot accurately forecast the value of acquiring multiple features together. Thus, it can produce sub-optimal policies. Second, the greedy strategy assumes that features can be acquired sequentially and the value of a feature can be observed before acquiring the next one. This assumption, however, might not be very practical. For example, doctors typically

order batches of measurements simultaneously such as blood count, cholesterol level, etc. For these reasons, we need to be able to reason with sets of features.

Reasoning with sets of features, however, poses tractability challenges. First of all, the number of subsets is exponential in the size of the feature set. Second, judging the value of acquiring a set of features requires taking an expectation over the possible values of the features in the set, which is also exponential in the number of the features. The good news is that, we do not need to consider all possible subsets in practice; certain features can render other features useless, while some features are useful only if acquired together. For example, an X-Ray might render a skin test useless for diagnosing tuberculosis. Similarly, a chest pain alone might not be useful for differentiating between cold and a heart disease; it becomes useful only if it is combined with other features, such as blood test.

In this chapter, we describe a data structure that discovers and exploits these types of constraints from the underlying probability distribution to reduce the search space. We propose *Value of Information Lattice* (`VOILA`) that reduces the space of all possible subsets by exploiting the constraints between the features. Additionally, `VOILA` makes it possible to share computations between different feature sets to reduce the computation time.

This chapter is organized as follows. We describe the notation and problem formulation in Section 3.2. We describe how we can reduce the search space and share computations using `VOILA` in Section 3.3. We show experimental results in Section 3.4 and conclude in Section 3.5.

3.2 Notation and Problem Formulation

Our main task is to classify a given instance with missing features and incur the minimum acquisition and misclassification costs. Let the instance be described by a set of features $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ and let Y be the random variable representing its class. We assume that the joint probability distribution $P(Y, \mathbf{X})$ is given and concern ourselves with feature acquisition during only inference (the conditional distribution $P(Y|\mathbf{X})$ is not appropriate, as most features are assumed to be unobserved). For the purpose of this chapter, we assume that we are given a Bayesian network, but any joint probabilistic model that allows us to efficiently answer conditional independence queries can be used.

In the notation, a bold face letter represents a set of random variables and non-bold face letter represents a single random variable. For example $\mathbf{X}$ represents the set of features, whereas $X_i \in \mathbf{X}$ represents a feature in $\mathbf{X}$ and Y represents the class variable. Additionally, a capital letter represents a random variable, and a lowercase letter represents a particular value of that variable; this applies to both individual variables and sets of variables. For example, Y represents a variable, and y represents a particular value that Y can take.

In addition to the probabilistic model, we are also given the cost models that specify feature acquisition costs and misclassification costs. Formally, we assume that we have a feature acquisition cost function that given a subset of features, $\mathbf{S}$, and the set of features whose values are known (evidence) $\mathbf{E}$, returns a non-negative real number $C(\mathbf{S} \mid \mathbf{e})$, and a misclassification cost model that returns the

X_1	X_2	Y	$P(X_1,X_2,Y)$
T	T	T	0.144
T	T	F	0.036
T	F	T	0.168
T	F	F	0.252
F	T	T	0.020
F	T	F	0.180
F	F	T	0.020
F	F	F	0.180

Features	Cost
X_1	
$X_1 \mid X_2$ =T	5
$X_1 \mid X_2$ =F	
X_2	
$X_2 \mid X_1$ =T	10
$X_2 \mid X_1$ =F	
X_1, X_2	15

Misclassification Costs		Actual Label	
		T	F
Predicted Label	T	0	50
	F	50	0

Figure 3.1: Example configuration with two features X_1 and X_2 and class variable Y. The table from left to right represent: the joint probability distribution $P(X_1, X_2, Y)$, the feature costs, and the misclassification costs.

misclassification cost c_{ij} incurred when Y is assigned y_i when the correct assignment is y_j. With these cost functions, we can model non-static feature acquisition costs; that is, the cost of acquiring the feature X_i can depend on what has been acquired so far and what their values are ($\mathbf{e}$) as well what is acquired in conjunction with this feature ($\mathbf{S} \setminus \{X_i\}$). The misclassification cost model does not assume symmetric costs; we can assign different costs for the false positives and false negatives.

Figure 3.1 shows a simple example configuration with two features, X_1 and X_2, and the class variable Y. In this simple example, the joint distribution $P(\mathbf{X}, Y)$ is represented as a table, the feature costs are simple independent costs for X_1 and X_2, and the misclassification cost is symmetric where both types of misclassifications cost the same and correct classification does not cost anything.

A diagnostic policy π is a decision tree whose leaf nodes represent classification decisions and non-leaf nodes represent feature that need to be acquired. Each path $\mathbf{p_s} \in \pi$ represents a sequence of ordered feature values $\mathbf{s}$ and a final classification

decision. We will often use $\mathbf{p_s}$ to represent an ordered version of $\mathbf{s}$. When the order of the variables in $\mathbf{s}$ is not important, then we will use either exchangeably. Typically, the order will be important for computing the feature costs, as the cost of a feature can depend on the values of previously acquired features, and the order will not be important for computing probabilities. An example conditional policy using the example configuration of Figure 3.1 is given in Figure 3.2. Each path $\mathbf{p} \in \pi$ has an associated feature cost and misclassification cost. The feature cost of a path can be computed as follows:

$$FC(\mathbf{p_s}) = \sum_{j=1}^{n} C(\mathbf{p_s}[j] \mid \mathbf{p_s}[1:j])$$

where $\mathbf{p_s}[j]$ represents the j^{th} feature in $\mathbf{p_s}$ and $\mathbf{p_s}[1:j]$ represents feature values 1 through j in $\mathbf{p_s}$. That is, the cost of acquiring the feature at depth i of the path depends on the values of the features acquired up until that depth. The expected misclassification cost of a path is:

$$EMC(\mathbf{p_s}) = EMC(\mathbf{s}) = \min_{y_i} \sum_{y_j} P(Y = y_j \mid \mathbf{s}) \times c_{ij} \tag{3.1}$$

The total cost of a path is the sum of the feature costs and the expected misclassification cost:

$$TC(\mathbf{p_s}) = FC(\mathbf{p_s}) + EMC(\mathbf{p_s})$$

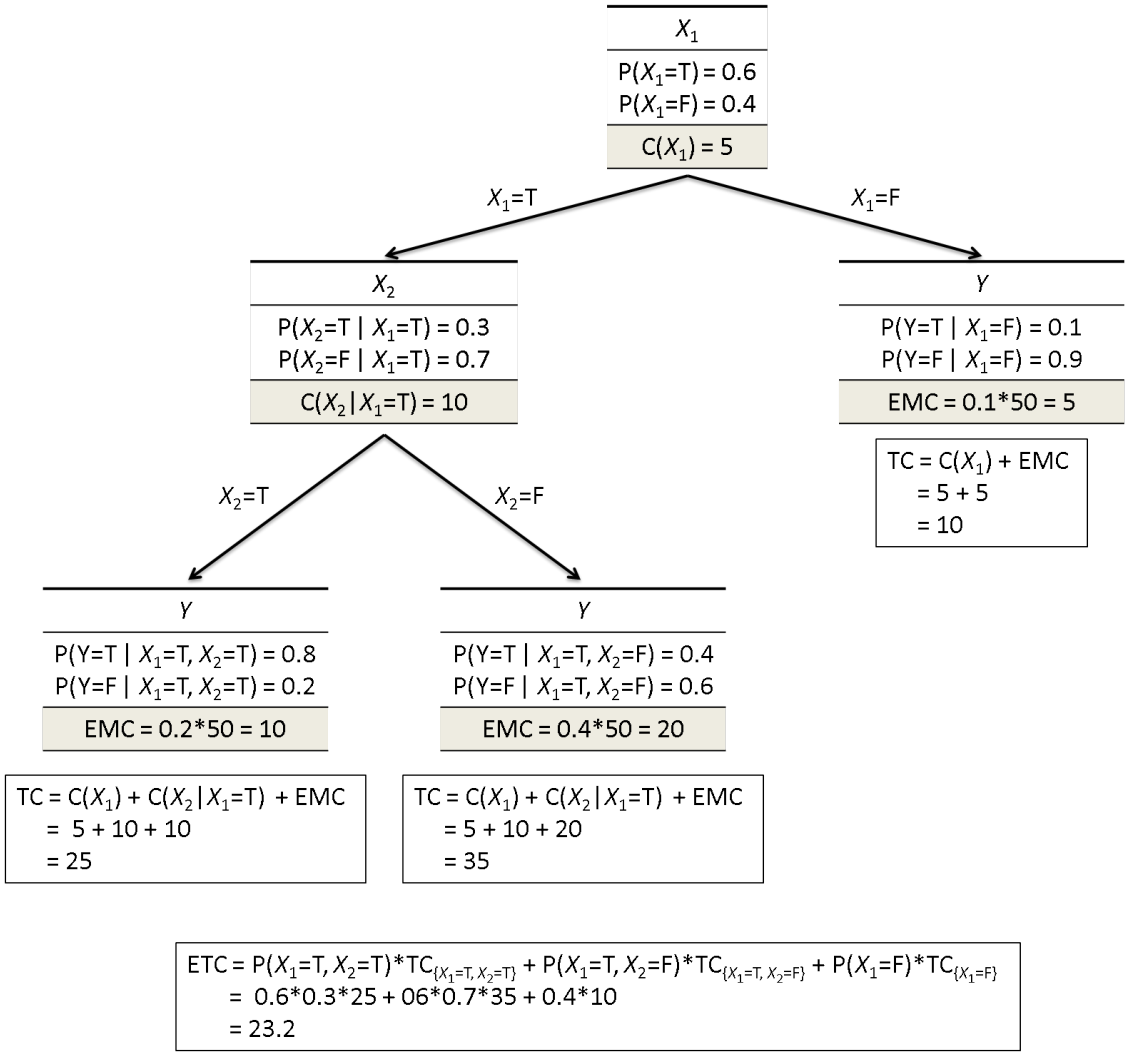


Figure 3.2: An example conditional policy with features X_1, X_2 and class variable Y. Each non-leaf node represents a feature acquisition, with probability distribution of the possible values, and the cost of the feature. Each path (e.g., $X_1 = T, X_2 = T$) has an acquisition cost and expected misclassification cost. The policy overall has an expected total cost ETC, which is the sum of total costs of each path, weighted by the probability of following that path.

The expected total cost of a policy is then the sum of the total cost of each path, weighted by the probability of following that path:

$$ETC(\pi) = \sum_{\mathbf{p_s} \in \pi} P(\mathbf{s}) TC(\mathbf{p_s}) \tag{3.2}$$

The objective of feature acquisition during inference is, given the joint probabilistic model and the cost models for acquisition and misclassification, find the policy that has the minimum expected total cost. However, building the optimal decision tree is known to be NP-complete (Hyafil and Rivest, 1976). Thus, most research have been greedy choosing the best feature that reduces the misclassification costs the most and has the lowest cost (e.g., Gaag and Wessels (1993); Dittmer and Jensen (1997)) or have developed heuristic feature scoring techniques (e.g., Núñez (1991); Tan (1990)).

In the greedy strategy, each path of the policy is extended with features that reduce the misclassification cost the most and that have the lowest cost. More specifically, the path $p_{\mathbf{s}}$ is replaced with new paths $\mathbf{p}_{\mathbf{s}\cup x_i^1}, \mathbf{p}_{\mathbf{s}\cup x_i^2}, \ldots, \mathbf{p}_{\mathbf{s}\cup x_i^n}$ where $x_i^1, x_i^2, \ldots, x_i^n$ are the values that X_i can take and X_i is the feature that has the highest *benefit*. We define the *benefit* of a feature X_i given a path $\mathbf{p}_{\mathbf{s}}$ as the reduction in the total cost of the path when the path is expanded with the possible values of X_i. More formally,

$$
\begin{aligned}
Benefit(X_i \mid \mathbf{p}_{\mathbf{s}}) &\triangleq TC(\mathbf{p}_{\mathbf{s}}) - \sum_{j=1}^{n} P(x_i^j \mid \mathbf{s}) TC(\mathbf{p}_{\mathbf{s}\cup x_i^j}) \\
&= FC(\mathbf{p}_{\mathbf{s}}) + EMC(\mathbf{s}) - \sum_{j=1}^{n} P(x_i^j \mid \mathbf{s}) \left(FC(\mathbf{p}_{\mathbf{s}\cup x_i^j}) + EMC(\mathbf{s}\cup x_i^j) \right) \\
&= FC(\mathbf{p}_{\mathbf{s}}) - \left(\sum_{j=1}^{n} P(x_i^j \mid \mathbf{s}) FC(\mathbf{p}_{\mathbf{s}\cup x_i^j}) \right) + EMC(\mathbf{s}) - \sum_{j=1}^{n} P(x_i^j \mid \mathbf{s}) EMC(\mathbf{s}\cup x_i^j) \\
&= FC(\mathbf{p}_{\mathbf{s}}) - (FC(\mathbf{p}_{\mathbf{s}}) + C(X_i \mid \mathbf{s})) + EMC(\mathbf{s}) - \sum_{j=1}^{n} P(x_i^j \mid \mathbf{s}) EMC(\mathbf{s}\cup x_i^j) \\
&= -C(X_i \mid \mathbf{s}) + EMC(\mathbf{s}) - \sum_{j=1}^{n} P(x_i^j \mid \mathbf{s}) EMC(\mathbf{s}\cup x_i^j)
\end{aligned}
$$

Note that, the last two terms are equivalent to the definition of expected value of information (Howard, 1966):

$$EVI(X_i \mid \mathbf{s}) = EMC(\mathbf{s}) - \sum_{j=1}^{n} P(x_i^j \mid \mathbf{s}) EMC(\mathbf{s} \cup x_i^j) \tag{3.3}$$

Substituting EVI, the definition of benefit becomes very intuitive:

$$Benefit(X_i \mid \mathbf{p_s}) = Benefit(X_i \mid \mathbf{s}) = EVI(X_i \mid \mathbf{s}) - C(X_i \mid \mathbf{s}) \tag{3.4}$$

With this definition, the greedy strategy iteratively finds the feature that has the highest positive benefit (value cost difference), acquires it, and stops acquisition when the benefit of acquiring a feature is non-positive.

We also note that it is straightforward to define EVI and $Benefit$ for a set $\mathbf{S}'$ of features just like we did for a single feature. The only difference is that the expectation needs to be taken over all members of the set $\mathbf{S}'$.

$$EVI(\mathbf{S}' \mid \mathbf{s}) = EMC(\mathbf{s}) - \sum_{\mathbf{s}'} P(\mathbf{s}' \mid \mathbf{s}) EMC(\mathbf{s} \cup \mathbf{s}') \tag{3.5}$$

and,

$$Benefit(\mathbf{S}' \mid \mathbf{s}) = EVI(\mathbf{S}' \mid \mathbf{s}) - C(\mathbf{S}' \mid \mathbf{s}) \tag{3.6}$$

However, there are a few problems with the greedy strategy. First, it is short-sighted. There exist sets $\mathbf{S} \subseteq \mathbf{X}$ such that $Benefit(\mathbf{S}) > \sum_{X_i \in \mathbf{S}} Benefit(X_i)$. This is easier to see, for example, for the `XOR` function, $Y = X_1$ `XOR` X_2, where X_1 and X_2

alone are not useful but they are determinative together. Due to this relationship, a greedy policy is not guaranteed to be optimal. Moreover, the greedy policy can prematurely stop acquisition because no single feature seems to provide positive benefit.

The second problem with the greedy strategy is that we often need to acquire a set of features simultaneously. For example, a doctor orders a set of lab tests when s/he sends the patient to lab, such as blood count, cholesterol level, etc. rather than ordering a single test, waiting for its result and ordering the next one. However, the traditional greedy strategy cannot handle reasoning with sets of features.

We would like to be able to reason with sets of features for these two reasons. Our objective in this chapter is, given an existing potentially empty set of already observed features $\mathbf{E}$ and their observed values $\mathbf{e}$, find the set that has the highest benefit:

$$\mathbf{L}(\mathbf{X}, \mathbf{S} \mid \mathbf{e}) \triangleq \underset{\mathbf{S} \subseteq \mathbf{X} \setminus \mathbf{E}}{\operatorname{argmax}} \mathit{Benefit}(\mathbf{S} \mid \mathbf{e}) \tag{3.7}$$

There are two problems with this formulation: first the number of subsets of $\mathbf{X} \setminus \mathbf{E}$ is exponential in the size of $\mathbf{X} \setminus \mathbf{E}$, and second, for each set $\mathbf{S}$, we need to take an expectation over all possible values for all features in the set. We address these two problems using a data structure which we describe next.

3.3 Value of Information Lattice (VOILA)

VOILA makes reasoning with sets of features tractable by reducing the space of possible sets and allowing EVI computation sharing between different sets. In this

section, we will first explain how we can reduce the space and then explain ways of computation sharing.

3.3.1 Reducing the Space of Possible Sets

In most domains, there are often complex interactions between the features and the class label. Contrary to the Naive Bayes assumption, features are often not conditionally independent given the class label. Some features are useless once some other features are already acquired. For example a chest X-Ray is typically more determinative than a skin test for tuberculosis. Similarly, some features are useless alone unless they are accompanied with other features. For example, a chest pain alone might be due to a variety of sicknesses; if it is accompanied with high cholesterol, it could mean a heart disease, whereas if it is combined with fever, it could mean cold. These types of interactions between the features put constraints on what the candidate sets could be.

Remember that we have assumed that we already have a joint probabilistic model over the features and the class variable, $P(Y, \mathbf{X})$. We will find these two types of feature interactions by asking probabilistic independence queries using $P(Y, \mathbf{X})$. Specifically, we assumed that we have a Bayesian network that represents $P(Y, \mathbf{X})$ and Bayesian networks allow us to find these types of interactions through standard d-separation algorithms.

Definition 1 *A set* $\mathbf{S} \subseteq \mathbf{X} \backslash \mathbf{E}$ *is* **irreducible** *with respect to evidence* $\mathbf{e}$ *iff* $\forall X_i \in \mathbf{S}$, X_i *is* <u>not</u> *conditionally independent of* Y *given* $\mathbf{e}$ *and* $\mathbf{S} \setminus \{X_i\}$.

Given a Bayesian network over $\mathbf{X}$ and Y, it is straightforward to check this d-separation property (Pearl, 1988).

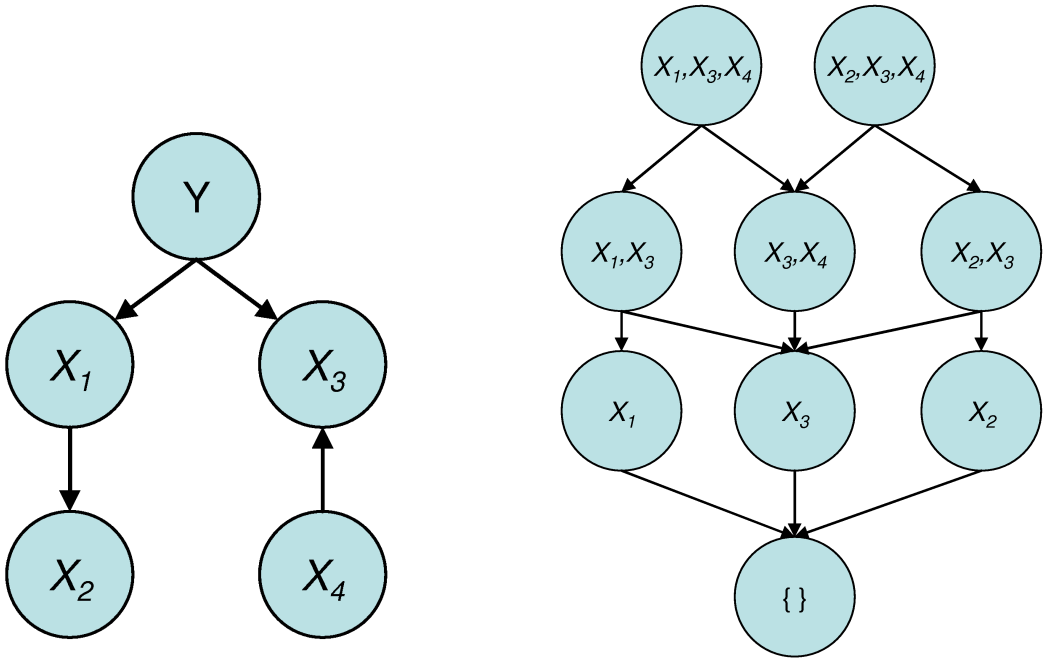
Proposition 1 *Let* $\mathbf{S}'$ *be a maximal irreducible subset of* $\mathbf{S}$ *with respect to* $\mathbf{e}$. *Then,* $EVI(\mathbf{S} \mid \mathbf{e}) = EVI(\mathbf{S}' \mid \mathbf{e})$.

Proof: Let $\mathbf{S}'' = \mathbf{S} \setminus \mathbf{S}'$. If $\mathbf{S}'$ is a maximal irreducible set, $\mathbf{S}' \cup \mathbf{E}$ d-separates Y and $\mathbf{S}''$. Otherwise, we could make $\mathbf{S}'$ larger by including the non-d-separated element(s) from $\mathbf{S}''$ in $\mathbf{S}'$. Thus, we have $P(Y \mid \mathbf{e}, \mathbf{s}) = P(Y \mid \mathbf{e}, \mathbf{S}', \mathbf{S}'') = P(Y \mid \mathbf{e}, \mathbf{S}')$. Substitution in Equations 3.1 and 3.5 yields the desired property.

Note that under the assumption that $C(\mathbf{S}' \mid \mathbf{e}) \leq C(\mathbf{S} \mid \mathbf{e})$ for any $\mathbf{S}' \subseteq \mathbf{S}$, it suffices to consider only the irreducible sets to find the optimal solution to the objective function in Equation (3.7). `VOILA` is a data structure that contains only the irreducible feature subsets of $\mathbf{X}$, with respect to a particular set of evidence $\mathbf{e}$. We next define `VOILA` formally.

Definition 2 *A* `VOILA` $\mathbf{V}$ *is a directed graph in which there is a node corresponding to each possible irreducible set of features, and there is a directed edge from a feature set* $\mathbf{S}$ *to each node which corresponds to a direct (maximal) subset of* $\mathbf{S}$. *Other subset relationships in the lattice are then defined through the directed paths in* $\mathbf{V}$.

Figure 3.3(a) shows a simple Bayesian network and its corresponding `VOILA`, with respect to the empty evidence set, is shown in Figure 3.3(b). Notice that the `VOILA` contains only the irreducible subsets given the Bayesian network with no evidence; for instance, the `VOILA` does not contain sets that include both X_1



(a) (b)

Figure 3.3: (a) A simple Bayesian network illustrating dependencies between attributes and the class variable. (b) The `VOILA` corresponding to the network.

and X_2 because X_1 d-separates X_2 from Y. We also observe that the number of irreducible subsets is 9 in contrast to $2^4 = 16$ possible subsets. Moreover, note that the largest subset size is now 3 in contrast to 4. Having smaller feature sets sizes has a dramatic effect on the value of information calculations. In fact, these savings can make solving the objective function optimally (Equation (3.7)) feasible in practice.

3.3.2 EVI Computation Sharing

Find the set $\mathbf{S}$ that has the highest $Benefit$ (Equation 3.6) requires computing $EVI(\mathbf{S})$ (Equation 3.5). However, computing $EVI(\mathbf{S})$ requires taking an expectation over all possible values of the features in $\mathbf{S}$. Moreover, searching the best set

among all the irreducible sets requires us to compute EVI for all irreducible sets. To make such computations tractable in practice, `VOILA` allows computation sharing between its nodes. In this chapter, we describe three possible ways of sharing computations between the nodes of `VOILA`.

3.3.2.1 Subset Relationships

`VOILA` exploits the subset relationships between different feature sets in order to avoid computing EVI for some nodes. First of all, if there is a directed path from node $\mathbf{S}_1$ to $\mathbf{S}_2$ in `VOILA`, then $S_1 \supset S_2$ and thus $EVI(\mathbf{S}_1 \mid \mathbf{e}) \geq EVI(\mathbf{S}_2 \mid \mathbf{e})$. Now assume that there is a directed path from $\mathbf{S}_i$ to $\mathbf{S}_j$ and $EVI(\mathbf{S}_i \mid \mathbf{e}) = EVI(\mathbf{S}_j \mid \mathbf{e})$. Then, all of the nodes on this path will also have the same EVI, thus we do not need to do the computation for those subsets. An algorithm that makes use of this observation is given in Algorithm 3.1.

Algorithm 3.1: Efficient EVI computation using `VOILA`.

Input: `VOILA` $\mathbf{V}$ and current evidence $\mathbf{E}$
Output: `VOILA` updated with correct EVI values

```
1  for all root node(s) S
2      value ← EVI(S | e)
3      ub(descendants(S)) ← value
4  for all leaf node(s) S
5      value ← EVI(S | e)
6      lb(ancestors(S)) ← value
7  for all node S where lb(S) ≠ ub(S)
8      value ← EVI(S | e)
9      lb(ancestors(S)) ← value
10     ub(descendants(S)) ← value
```

In order to share computations between different nodes of the lattice, we keep lower and upper bounds on the EVI of a node. The lower bound is determined by the values of the descendants of the node whereas the upper bound is determined by the values of its ancestors. First, we initialize these bounds by computing the value of the information at the boundary of the lattice, i.e., the root node(s) and the leaf node(s) (lines 1–6). Then, we loop over the nodes whose upper bounds and lower bounds are not equal (line 7–10), computing their values and updating the bounds at their ancestors and descendants. The order in which to choose the nodes in line 7 so that the number of sets for which a value is calculated is minimum is still an open question. A possible heuristic is choosing a middle node on a path between two nodes for which the values have already been calculated.

3.3.2.2 Information Pathways at the Underlying Bayesian Network

The second mechanism that `VOILA` has to share EVI computations is through the edges of the underlying Bayesian network. We specifically make use of the following fact:

Proposition 2 *For all* $\mathbf{S}_1$ *and* $\mathbf{S}_2$*, if* $\mathbf{S}_1$ *d-separates* Y *from* $\mathbf{S}_2$ *with respect to* $\mathbf{e}$*, then* $EVI(\mathbf{S}_1 \mid \mathbf{e}) \geq EVI(\mathbf{S}_2 \mid \mathbf{e})$.

Proof: Consider $\mathbf{S}_{12} = \mathbf{S}_1 \cup \mathbf{S}_2$. Because of the subset relationship, we know that $EVI(\mathbf{S}_{12} \mid \mathbf{e}) \geq EVI(\mathbf{S}_1 \mid \mathbf{e})$ and $EVI(\mathbf{S}_{12} \mid \mathbf{e}) \geq EVI(\mathbf{S}_2 \mid \mathbf{e})$.

$$\begin{aligned}
EVI(\mathbf{S}_{12} \mid \mathbf{e}) &= EMC(Y \mid \mathbf{e}) - \sum_{\mathbf{s}_{12}} P(\mathbf{s}_{12} \mid \mathbf{e}) EMC(Y \mid \mathbf{e}, \mathbf{s}_{12}) \\
&= EMC(Y \mid \mathbf{e}) - \sum_{\mathbf{s}_1} \sum_{\mathbf{s}_2} P(\mathbf{s}_1, \mathbf{s}_2 \mid \mathbf{e}) EMC(Y \mid \mathbf{e}, \mathbf{s}_1, \mathbf{s}_2) \\
&= EMC(Y \mid \mathbf{e}) - \sum_{\mathbf{s}_1} \sum_{\mathbf{s}_2} P(\mathbf{s}_1, \mathbf{s}_2 \mid \mathbf{e}) EMC(Y \mid \mathbf{e}, \mathbf{s}_1) \\
&= EMC(Y \mid \mathbf{e}) - \sum_{\mathbf{s}_1} P(\mathbf{s}_1 \mid \mathbf{e}) EMC(Y \mid \mathbf{e}, \mathbf{s}_1) \\
&= EVI(\mathbf{S}_1 \mid \mathbf{e}) \\
&\geq EVI(\mathbf{S}_2 \mid \mathbf{e})
\end{aligned}$$

Corollary: The Markov Blanket of Y is the set that has the highest EVI in our search space, as it d-separates all of the remaining variables from Y.

These relationships can very well be exploited like we exploited the subset relationships above. Instead of just using the subset relationships, we can use both subset and independence relationships. One simple way to make use of Algorithm 3.1 without modification is to add edges between any $\mathbf{S}_1$ and $\mathbf{S}_2$ where the independence property holds. An example $\mathbf{S}_1$ and $\mathbf{S}_2$ according to our toy network in Figure 3.3(a) would be $\mathbf{S}_1 = \{X_1\}$ and $\mathbf{S}_2 = \{X_2\}$. Thus, we can add a directed edge from X_1 to X_2 in our `VOILA` in Figure 3.3(b) and Algorithm 3.1 will work just fine.

3.3.2.3 Incremental Inference

The third and the last mechanism that VOILA uses for computation sharing is through caching of probabilities at its nodes. For each candidate set $\mathbf{S} \in \mathbf{V}$, we need to compute $EVI(\mathbf{S} \mid \mathbf{e})$ which requires computing $P(\mathbf{S} \mid \mathbf{e})$ and $EMC(Y \mid \mathbf{S}, \mathbf{e})$. If we cache the conditional probabilities at each node of $\mathbf{V}$, then to compute $P(\mathbf{S} \mid \mathbf{e})$, we find one of its supersets $\mathbf{S}_i = \mathbf{S} \cup \{X_i\}$ and then compute $P(\mathbf{S} \mid \mathbf{e}) = \sum_{x_i} P(\mathbf{S}, X_i = x_i \mid \mathbf{e})$.

Computing $EMC(Y \mid \mathbf{S}, \mathbf{e})$ requires computing $P(Y \mid \mathbf{S}, \mathbf{e})$. To perform this computation efficiently, we cache the state of the junction tree at each node of the VOILA. Then, we find a subset, $\mathbf{S}_j$, such that $\mathbf{S} = \mathbf{S}_j \cup \{X_j\}$. We compute $P(Y \mid \mathbf{S}, \mathbf{e})$ by integrating the extra evidence to the junction tree at node $\mathbf{S}_j$ that is used to compute $P(Y \mid \mathbf{S}_j, \mathbf{e})$.

3.3.3 Constructing VOILA

Efficient construction of VOILA is not a straightforward task. The brute force approach would be to enumerate all possible subsets of $\mathbf{X} \setminus \mathbf{E}$ and for each subset check whether it is irreducible. However, this brute force approach is clearly impractical. Because the number of nodes in VOILA is expected to be much fewer than the number of possible subsets of $\mathbf{X} \setminus \mathbf{E}$, if we can be smart about which sets we consider for inclusion in $\mathbf{V}$, we can construct it more efficiently. That is, instead of generating all possible candidates and checking whether they are irreducible or not, we try to generate only irreducible sets. We first introduce the notion of a

dependency constraint and then explain how we can use dependency constraints to efficiently construct `VOILA`.

Definition 3 *A* dependency constraint *for a feature* $X_i \in \mathbf{S}$ *with respect to* $\mathbf{S}$ *and* $\mathbf{E}$ *is the constraint on* $\mathbf{S} \cup \mathbf{E}$ *required for a dependency between* X_i *and* Y *to exist.*

For instance, in our running example, a dependency constraint for X_2 is $\neg X_1$; in other words, in order for X_2 to be relevant, X_1 should not be included in $\mathbf{S} \cup \mathbf{E}$. Similarly, the dependency constraint for X_4 is X_3, meaning that X_3 must be included in $\mathbf{S} \cup \mathbf{E}$. Specifically, a dependency constraint for a feature X_i requires that all X_j on the path from Y to X_i not to be included in $\mathbf{S} \cup \mathbf{E}$ if X_j is not part of a v-structure; if X_j is part of a v-structure, then either X_j or one of its descendants must be included in $\mathbf{S} \cup \mathbf{E}$ (we refer to these latter constraints as *positivity constraints*). The algorithm that uses these ideas to compute the dependency constraints for each feature is given in Algorithm 3.2.

Algorithm 3.2: Dependency constraint computation for X_i.

Input: X_i, Y
Output: Dependency constraint for X_i, denoted $DC(X_i)$

```
1 DC(X_i) ← false
2 for each undirected path p_j between X_i and Y
3     DC_j(X_i) ← true
4     for each X_k on the path p_j
5         if X_k does not a cause a v-structure then
6             DC_j(X_i) ← DC_j(X_i) ∧ ¬X_k
7         else
8             DC_j(X_i) ← DC_j(X_i) ∧ (X_k ∨ Descendants(X_k))
9     DC(X_i) ← DC(X_i) ∨ DC_j(X_i)
```

We use the dependency constraints to check whether a set is irreducible. Because a set $\mathbf{S}$ is irreducible only if a dependency between all of its elements and Y

exists, the dependency constraint for the set **S** is the conjunction of the dependency constraints of its members. The irreducibility of **S** can be checked by setting the elements of **S** and **E** to "true" and setting the remaining elements of **X** to "false" and evaluating the set's dependency constraint. In our running example, the dependency constraint for the set $\{X_2, X_4\}$ is $\neg X_1 \wedge X_3$. Assuming $\mathbf{E} = \emptyset$, when we set the members of $\{X_2, X_4\}$ to true, and set the remaining features, X_1 and X_3, to false, $\neg X_1 \wedge X_3$ then evaluates to false and thus this set is not irreducible. This makes sense because given no evidence, X_4 is independent of Y, so while $\{X_2\}$ is a useful feature set to consider for acquisition, $\{X_2, X_4\}$ is not.

3.3.3.1 Construction Algorithm

We now describe constructing the `VOILA` using dependency constraints. `VOILA` construction proceeds in a bottom up fashion, beginning with the lowest level, which initially contains only the empty set and constructs new irreducible feature sets by adding one feature at a time into the `VOILA` structure. Algorithm 3.3 gives the details of the algorithm. The algorithm keeps track of the irreducible feature sets **IS**, and the set of potentially irreducible feature sets **PS**. A feature set is potentially irreducible if it is not irreducible but can be made reducible by adding more features into it. Note that this is possible due to the non-monotonic nature of d-separation.

The check for potential irreducibility can be done efficiently by setting members of **S** and **E** to true, setting literals with positivity constraints that remain to be processed to true, setting the remaining literals to false, and evaluating the set

Algorithm 3.3: The VOILA construction algorithm.

Input: Set of features $\mathbf{X}$ and class variable Y.
Output: The VOILA data structure $\mathbf{V}$, given $\mathbf{E}$.

```
1  Pick an ordering of elements of X = X_{i_1}, X_{i_2}, ..., X_{i_n}
2  IS ← {∅}; PS ← ∅
3  for j = 1 to n
4      for each S ∈ IS ∪ PS
5          S' ← S ∪ X_{i_j}; DC(S') ← DC(S) ∧ DC(X_{i_j})
6          if S' is irreducible then
7              IS ← IS ∪ {S'}; Add a node corresponding to S' to V
8          else
9              if S' is potentially irreducible then
10                 PS ← PS ∪ {S'}
11     Remove from PS all sets that are no longer potentially irreducible
12 max = size of largest S in IS; L_l = {S | S ∈ IS and |S| = l}
13 for l = 0 to max − 1
14     for each S ∈ L_l
15         for each S' ∈ L_{l+1}
16             if S ⊂ S' then
17                 Add an edge from S' to S to V
```

dependency constraint. The difference between checking for potential irreducibility and irreducibility is that we set positively constraint literals that might be added later to true for the potential irreducibility whereas we set them to false to check for irreducibility. When we are done processing feature X_{i_j}, we remove from **PS** any potentially irreducible set that cannot become irreducible because X_{i_j} will not be re-considered (line 11).

3.3.3.2 Analysis of VOILA Construction Algorithm

The construction algorithm puts a node in the VOILA only if the corresponding set is irreducible (lines 6 and 7). Moreover, by keeping track of potentially irreducible

sets (lines 8–10), we generate every possible irreducible set that can be generated. Thus, VOILA contains only and all of the possible irreducible subsets of $\mathbf{X}$.

The worst-case running time of the algorithm is still exponential in the number of initially unobserved features, $\mathbf{X} \setminus \mathbf{E}$, because number of irreducible sets can potentially be exponential. The running time in practice, though, depends on the structure of the Bayesian network that the VOILA is based upon and the ordering of the variables in line 1. The for loop at line 4 iterates over each irreducible and potentially irreducible sets that have been generated so far, and the number of potentially-irreducible sets generated depends on the ordering chosen. The number of irreducible sets, however, depends on the structure of the underlying Bayesian network, and we empirically show in the experimental results section that for five real world datasets, the number of irreducible subsets is substantially smaller than the number of possible subsets.

A good ordering processes features with literals with positivity constraints in other features' dependency constraints earlier. That is, for each undirected path from Y to X_i that includes X_j in a v-structure, a good ordering puts X_j earlier in the ordering than everything between X_j and X_i. For instance, in our sample Bayesian network in Figure 3.3(a), we should consider X_3 earlier than X_4. We refer to an ordering as *perfect* if it satisfies all the positivity constraints. If a perfect ordering is used, VOILA construction algorithm never generates a potentially irreducible set. Unfortunately, it is not always possible to find a perfect ordering. A perfect ordering is not possible when two features have each other as a positivity constraint literal in their dependency constraints. This case occurs only when there is a path from Y

to Y with two or more v-structures. A perfect ordering was possible in four of the five real world datasets that we used.

3.3.4 Using VOILA for Feature-value Acquisition

VOILA makes searching the space of all possible subsets tractable in practice. Using this flexibility, it is possible to devise several different acquisition policies. We describe two policies as example policies in this section.

The first acquisition policy aims to capture the practical setting where more than one feature is acquired at once. The policy can be constructed using VOILA as follows. Each path $\mathbf{p_s}$ of the policy π (which is initially empty) is repeatedly extended by acquiring the set $\mathbf{S'} \in \mathbf{V}$ that has the best $Benefit(\mathbf{S'} \mid \mathbf{s}, \mathbf{e})$. The policy construction ends when no path can be extended, i.e., all candidate sets have non-positive $Benefit$ values for each path of π.

The second acquisition policy adds a look-ahead capability to the greedy policy. That is, rather than repeatedly extending each path $\mathbf{p_s}$ of policy π with the feature X_i that has the highest $Benefit(X_i \mid \mathbf{s}, \mathbf{e})$, we add a look-ahead capability, and first find the set $\mathbf{S'} \in \mathbf{V}$ that has the highest $Benefit(\mathbf{S'} \mid \mathbf{s}, \mathbf{e})$. Then, instead of acquiring all features in $\mathbf{S}$ all at once like we did in the above policy, we find the feature $X_i \in \mathbf{S'}$ that has the highest $Benefit(X_i \mid \mathbf{s}, \mathbf{e})$ and acquire it to extend $\mathbf{p_s}$.

3.4 Experiments

We experimented with five real-world medical datasets that Turney (1995) described and used in his paper. These datasets are Bupa Liver Disorders, Heart Disease, Hepatitis, Pima Indians Diabetes, and Thyroid Disease, which are all available from the UCI Machine Learning Repository (Frank and Asuncion, 2010). The datasets had a varying number of features ranging from five to 20. Four out of five datasets had binary labels, whereas the Thyroid dataset had three labels.

For each dataset, we first learned a Bayesian Network that both provides the joint probability distribution $P(Y, \mathbf{X})$ and efficiently answers conditional independence queries thorough d-separation (Pearl, 1988). We built a `VOILA` for each dataset using the learned Bayesian Network. We first present statistics on each dataset, such as the number of features and number of nodes in the `VOILA`, and then compare various acquisition policies.

3.4.1 Search Space Reduction

Table 3.1 shows aggregate statistics about each dataset, describing the number of features, the number of all possible subsets, the number of subsets in `VOILA` and the percent reduction in the search space. As this table shows, the number of irreducible subsets is substantially less than all possible subsets. For the Thyroid Disease dataset, for example, the number of possible subsets is over a million, however the number of irreducible subsets is fewer than thirty thousand. This enormous

Table 3.1: Aggregate statistics about each dataset. The number of irreducible subsets, i.e., the number of nodes in `VOILA`, is substantially fewer than the number of all possible subsets.

Dataset	Features	All Subsets	Nodes in `VOILA`	Reduction
Bupa Liver Disorders	5	32	26	19%
Pima Indians Diabetes	8	256	139	46%
Heart Disease	13	8,192	990	88%
Hepatitis	19	524,288	18,132	97%
Thyroid Disease	20	1,048,576	28,806	97%

reduction in the search space makes searching through the possible sets of features tractable in practice.

3.4.2 Expected Total Cost Comparisons

We compared the expected total costs (Equation 3.2) of four different acquisition policies for each dataset. These policies are as follows:

- `No Acquisition`: This policy does not acquire any features; it aims to minimize the expected misclassification cost based on the prior probability distribution of the class variable, $P(Y)$.

- `Markov Blanket`: This policy acquires every useful feature, regardless of the misclassification costs. The Market Blanket of Y in a Bayesian network is defined as Y's parents, children, and its children's other parents (Pearl, 1988). Intuitively, it is the minimal set $\mathbf{S} \subseteq \mathbf{X}$ such that $Y \perp (\mathbf{X} \setminus \mathbf{S}) \mid \mathbf{S}$.

- `Greedy`: This policy repeatedly expands each path $\mathbf{p_s}$ of an initially empty policy π by acquiring the feature X_i that has the highest positive $Benefit(X_i \mid$

$\mathbf{s}$) (Equation 3.4). The policy construction ends when no path can be extended with a feature with a positive $Benefit$ value.

- `Greedy-LA`: This policy adds a look-ahead capability to the `Greedy` strategy. This policy repeatedly expands each path $\mathbf{p_s}$ of an initially empty policy π by first finding the set $\mathbf{S}'$ that has the highest positive $Benefit(\mathbf{S}' \mid \mathbf{s})$ (Equation 3.6) and then acquiring the feature $X_i \in \mathbf{S}'$ that has the maximum $Benefit(X_i \mid \mathbf{s})$ (Equation 3.4). The policy construction ends when no set with a positive $Benefit$ value can be found for any path of the policy.

The feature costs for each dataset are described in detail by Turney (1995). In summary, each feature can either have an independent cost, or can belong to a group of features, where the first feature in that group incurs an additional cost. For example, the first feature from a group of blood measurements incurs the overhead cost of drawing blood from the patient. The feature costs are based on the data from Ontario Ministry of Health (1992).

We observed that most of the features were assigned the same cost. For example, four out of five features in the Bupa Liver Disorders dataset, 13 out of 19 features in the Hepatitis dataset, six out of eight features in the Diabetes dataset, 16 out of 20 features in the Thyroid Disease dataset were assigned the same cost. In addition to the given costs, which we call the real feature costs, we also experimented with randomly generated feature and group costs. For each feature, we randomly generated a cost between 1 and 100, and for each group we generated a

Table 3.2: Example misclassification cost matrix (c_{ij}) for the symmetric and asymmetric misclassification costs. c_{ij} are set in way to achieve a prior expected misclassification cost of 1. In the symmetric cost case, choosing the most probable class leads to $EMC = 1$, whereas, in the asymmetric cost case, the choosing either class is indifferent and both leads to the same EMC of 1.

Actual Class	Prior Probability	Pred. Class	Symm. Cost	Asymm. Cost
y_1	$P(y_1) = 0.6510$	y_1 y_2	0 2.866	0 2.866
y_2	$P(y_2) = 0.3490$	y_1 y_2	2.866 0	1.536 0

cost between 100 and 200. We repeated the experiments with three different seeds for each dataset.

The misclassification costs were not defined in (Turney, 1995). One reason could be that it is easier to define the feature costs, but defining the cost of a misclassification can be non-trivial. Instead, Turney (1995) tests different acquisition strategies using various misclassification costs. We follow a similar technique with a slight modification. We compare the above acquisition policies under both symmetric ($c_{ij} = c_{ji}$) and asymmetric misclassification costs. To be able to judge how the misclassification cost structure affects feature acquisition, we unify the presentation, and compare different acquisition strategies under the same a priori expected misclassification costs, as defined in Equation (3.1). Specifically, we compare the acquisition policies under various a priori EMC that are achieved by varying the c_{ij} accordingly. We show an example misclassification table for an EMC value of 1 in Table 3.2. For the real feature cost case, we varied the EMC between 0 and 2000, and varied it from 0 to 4000 for the synthetic feature cost case.

We compare the `Greedy`, `Greedy-LA`, and `Markov Blanket` policies by plotting how much cost each policy saves with respect to the `No Acquisition` policy. In the X axis of the plots, we vary the apriori expected misclassification cost using the methodology we described above. We plot the savings on the Y axis. For each dataset, we plot four different cases: the cross product of {symmetric, asymmetric} misclassification costs, and {real, synthetic} feature costs. It is important to note that we do not evaluate different acquisition policies on the datasets. Rather, we compute the expected total cost, ETC, as defined in Equation (3.2), considering all possible instances, $\mathbf{X}$, and weighting them based on the probability of seeing such an instance, $P(\mathbf{X})$.

The results for the Liver Disorders, Diabetes, Heart Disease, Hepatitis, and Thyroid Disease are given in Figures 3.4, 3.5, 3.6, 3.7, and 3.8 respectively. For each figure, symmetric misclassification cost scenarios are given in subfigures (a) and (c), whereas the asymmetric misclassification cost scenarios are presented in (b) and (d). Similarly, the real feature cost scenarios are given in (a) and (b) and the synthetic feature cost scenarios are presented in (c) and (d). We next summarize the results.

- We found that the `Greedy` policy often prematurely stops acquisition, performing even worse than the `Markov Blanket` strategy. This is true for most of the datasets, regardless of the feature and misclassification cost structures. The fact that the `Greedy` strategy can perform worse than `Markov Blanket` strategy is really troubling. However, `Greedy-LA` never performs worse than `Markov Blanket` as expected.

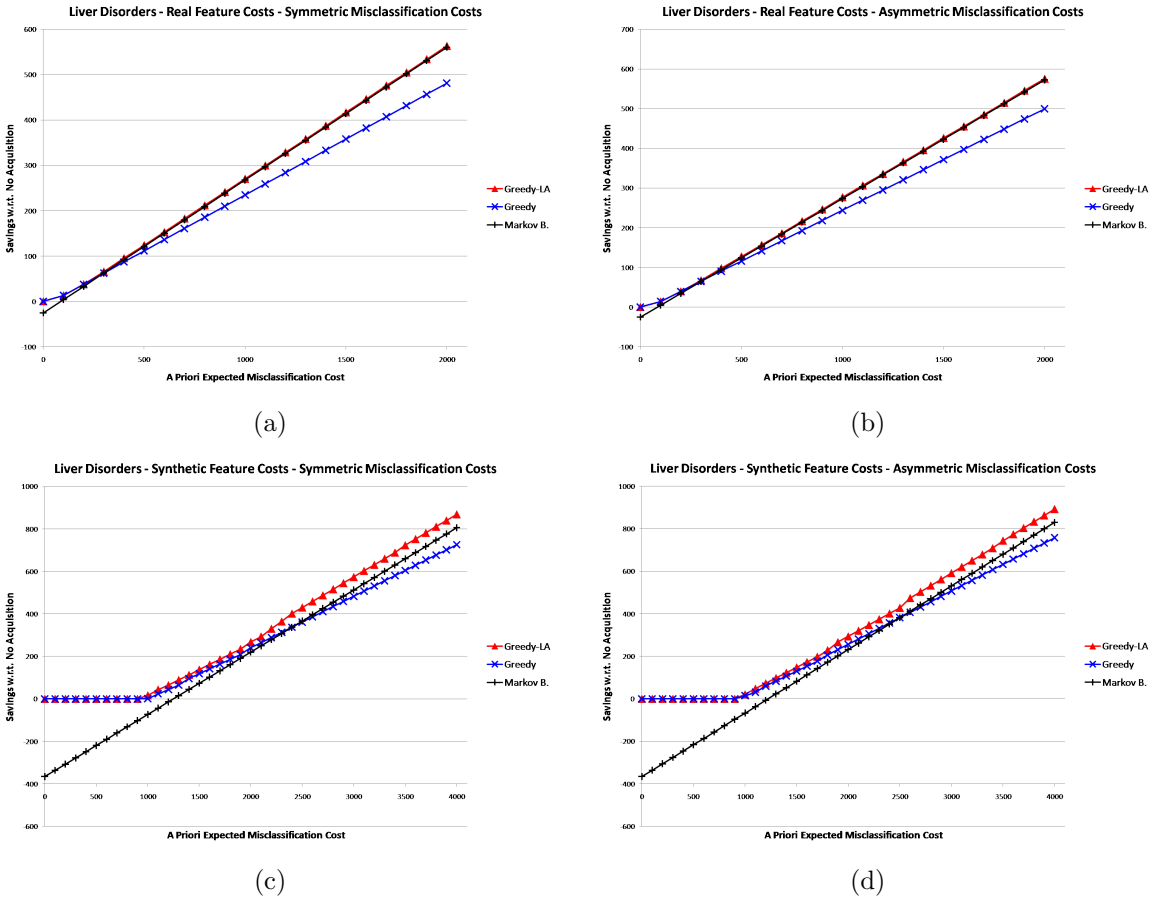


Figure 3.4: Expected Total Cost (ETC) comparisons for the Bupa Liver Disorders dataset. The apriori class distribution is as follows: $P(Y) = [0.4959, 0.5041]$.

- `Greedy-LA` strategy never performs worse than any other strategy under any setting.

- The misclassification cost structure (symmetric or asymmetric) had a considerable affect on how the policies behaved. The differences between symmetric and asymmetric cases were particularly evident for datasets where the class distribution was more imbalanced, such as the Diabetes (Figure 3.5), Hepatitis (Figure 3.7), and the Thyroid Disease (Figure 3.8) datasets. The differences due to the misclassification cost structure can be summarized as follows:

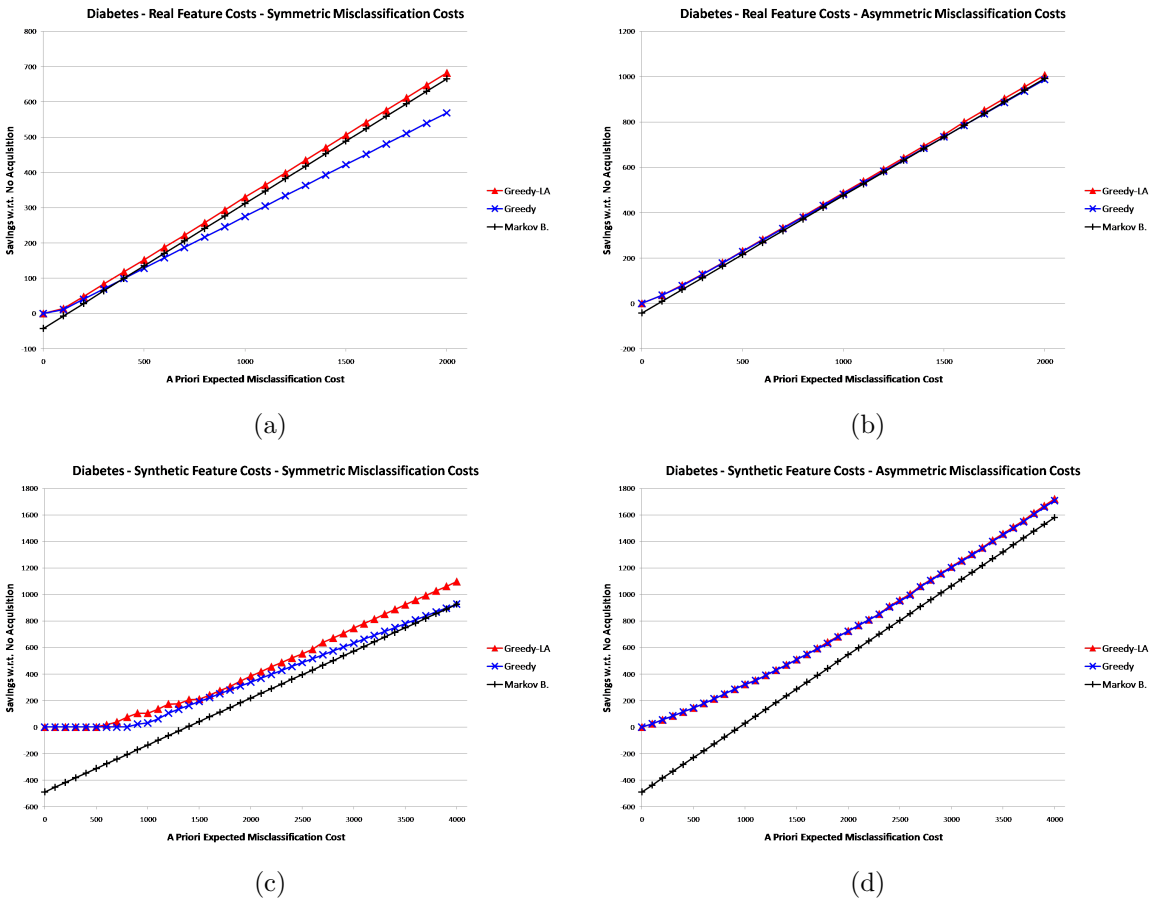


Figure 3.5: Expected Total Cost (ETC) comparisons for the Pima Indian Diabetes dataset. The apriori class distribution is as follows: $P(Y) = [0.6510, 0.3490]$.

– When the class distribution is imbalanced and the misclassification cost is symmetric, acquiring more information cannot change the classification decisions easily due to the class imbalance, thus the features do not have high EVI values. On the other hand, if the misclassification costs are asymmetric, features tend to have higher EVI values. Thus, the `Greedy` and `Greedy-LA` strategies start acquiring features earlier in the X axis for the asymmetric cases compared to their symmetric counterparts. For example, for the Thyroid disease dataset, the `Greedy` strategy starts acquisition only when the EMC is greater than 600 for real fea-

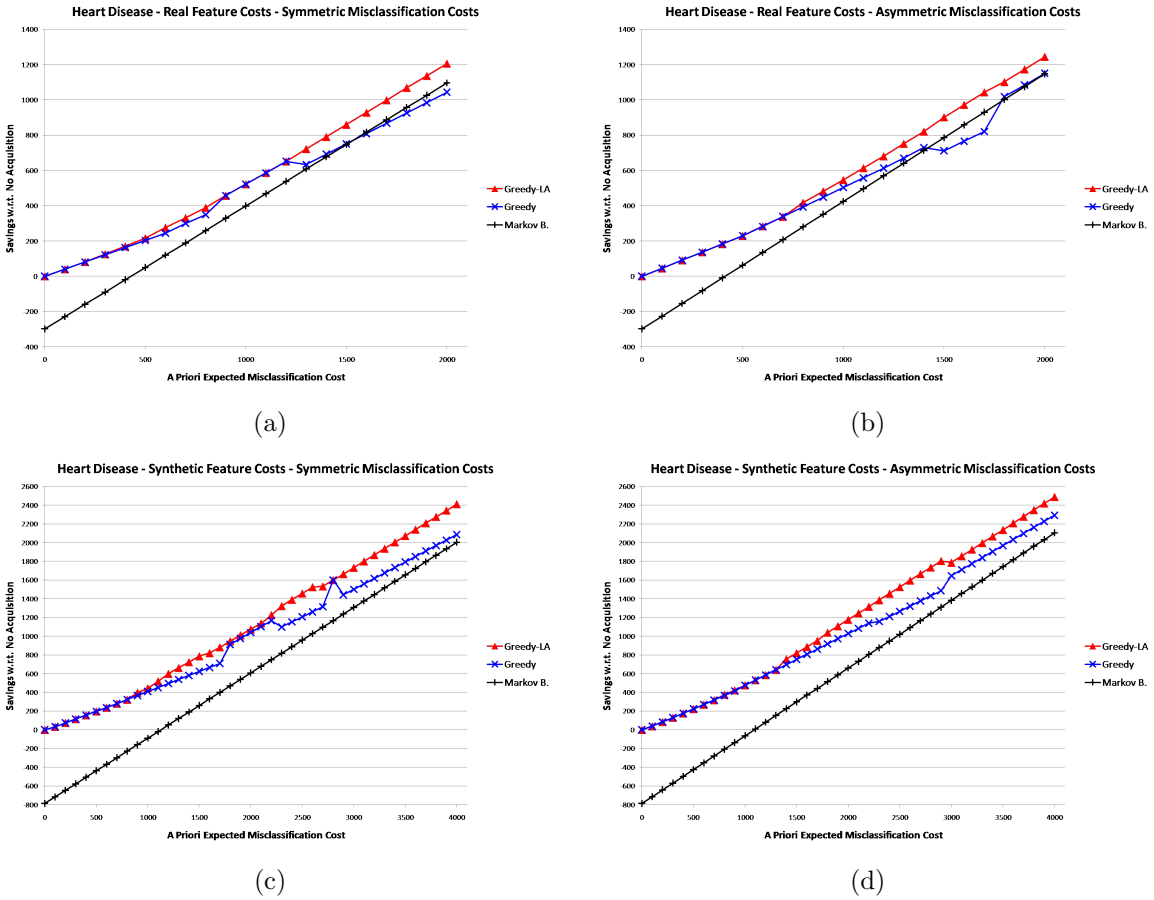


Figure 3.6: Expected Total Cost (ETC) comparisons for the Heart Disease dataset. The apriori class distribution is as follows: $P(Y) = [0.5444, 0.4556]$.

ture costs and symmetric misclassification costs (Figure 3.8(a)); however, it starts acquiring when the EMC reaches only 100 for the asymmetric case (Figure 3.8(b)). For the synthetic feature costs, the results are more dramatic; neither `Greedy` nor `Greedy-LA` acquires any features for the symmetric cost case (Figure 3.8(c)), whereas they start acquisition when $EMC = 200$ for the asymmetric case (Figure 3.8(d)).

– In the same realm with the above results, the slope of the savings for the asymmetric case is much higher for the asymmetric misclassification costs compared to the symmetric case.

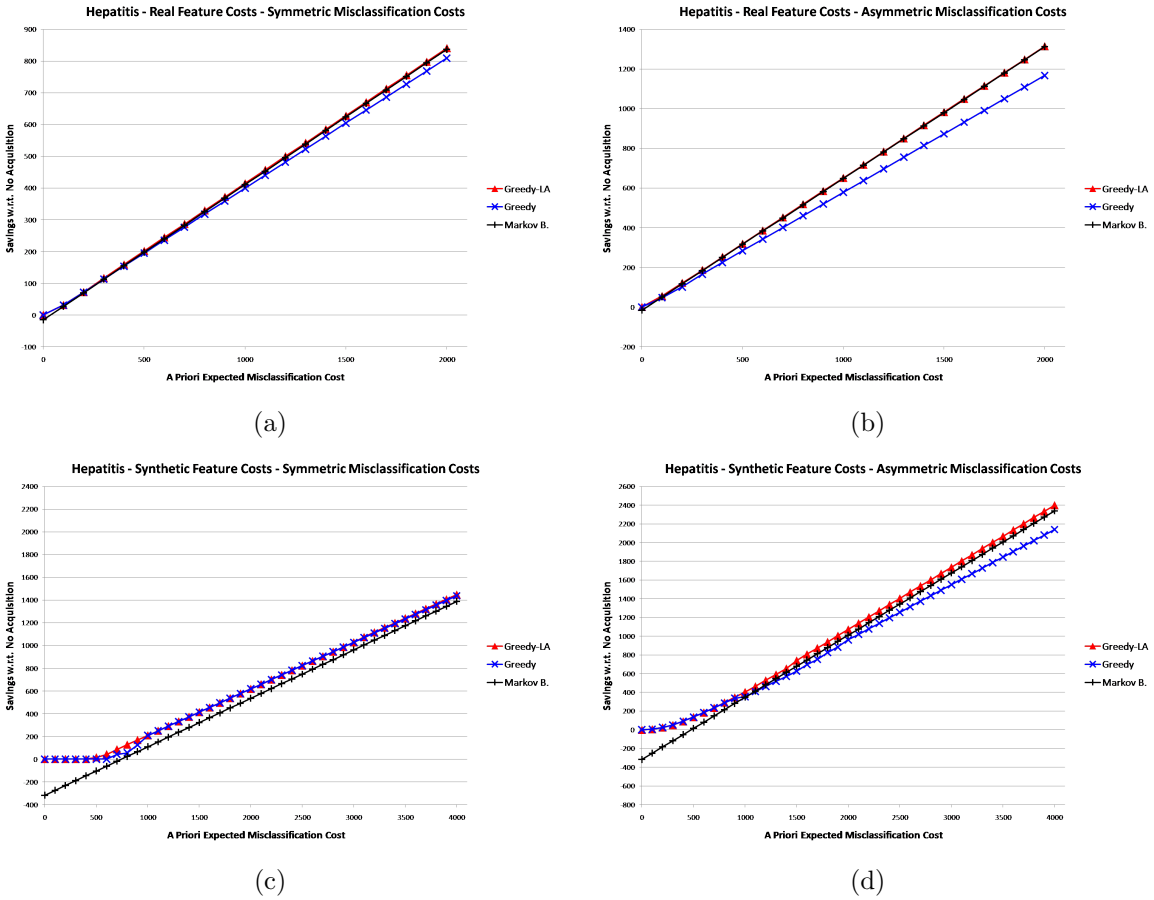


Figure 3.7: Expected Total Cost (ETC) comparisons for the Hepatitis dataset. The apriori class distribution is as follows: $P(Y) = [0.7908, 0.2092]$.

  - The misclassification cost structure causes differences between the `Greedy` and `Greedy-LA` policies in a few cases. For the Diabetes dataset `Greedy` policy performs worse when the misclassification costs are symmetric (Figures 3.5(a) and 3.5(c)), whereas for the Hepatitis dataset, `Greedy` performs worse for the asymmetric misclassification costs (Figures 3.7(b) and 3.7(d)).

- The `Greedy` policy sometimes has an erratic, unpredictable, and unreliable performance as the expected misclassification changes. It possibly hits a local

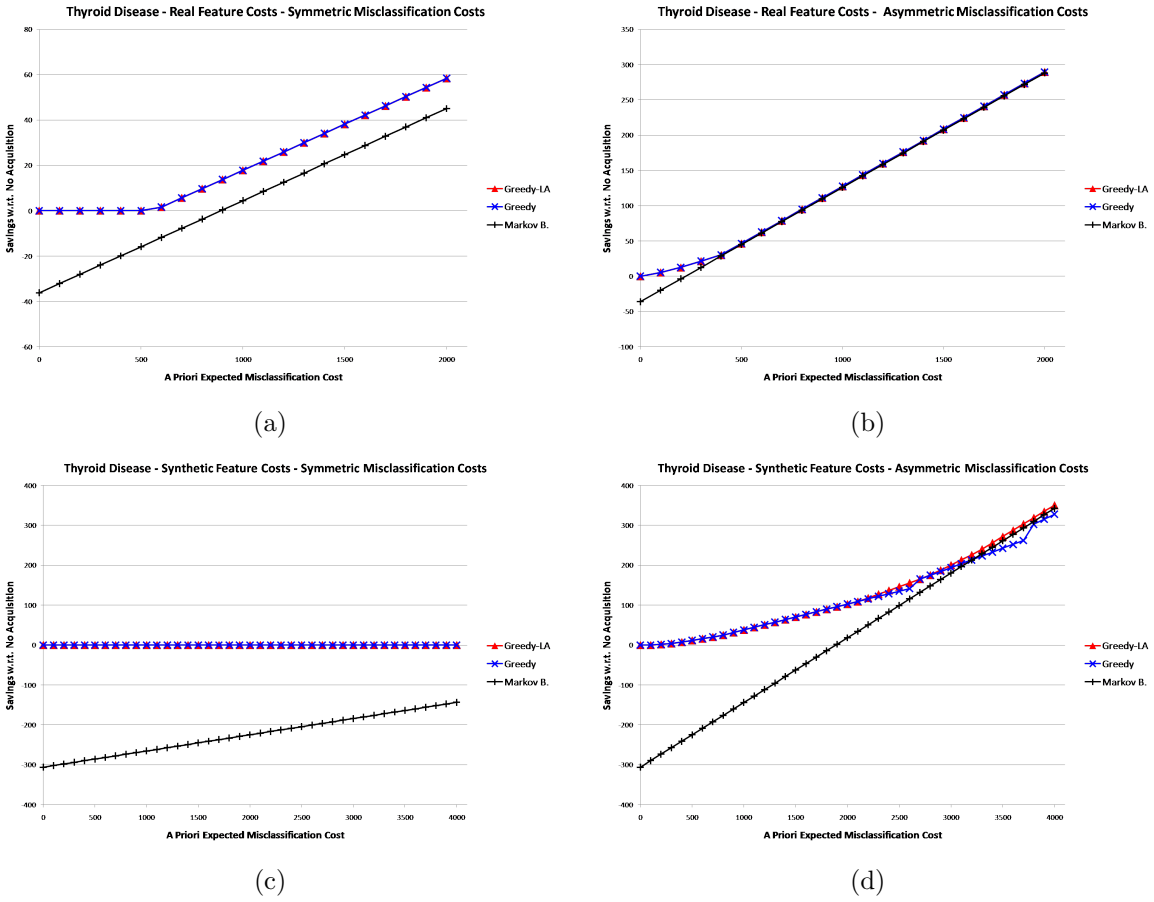


Figure 3.8: Expected Total Cost (ETC) comparisons for the Thyroid Disease dataset. The apriori class distribution is as follows: $P(Y) = [0.0244, 0.0507, 0.9249]$.

minima, gets out of it later, and hits local minima again (Figures 3.6 and 3.8(d)).

We finally present an aggregate summary of the results in Table 3.3. Table 3.3 shows how much the `Greedy` policy and the `Greedy-LA` policy saves over the `Markov Blanket` policy. The results are presented as the average saving over various intervals, such as [0-500). As this table also shows, the `Greedy-LA` policy never loses compared to the `Markov Blanket` policy, as one would expect. Additionally, the `Greedy-LA` policy wins over the `Greedy` policy for most of the cases,

Table 3.3: Savings of `Greedy` (`GR`) and `Greedy-LA` (`LA`) with respect to the `Markov Blanket` policy, averaged over different intervals. An entry is in bold if it is worse than `Greedy-LA`, and it is in red if it is worse than `Markov Blanket`.

	Liver		Diabetes		Heart		Hepatitis		Thyroid	
	GR	LA	GR	LA	GR	LA	GR	LA	GR	LA
Real Feature Costs & Symmetric Misclassification Costs										
[0-500)	**6.77**	9.08	**15.49**	24.27	**240.59**	243.31	**4.19**	5.86	28.07	28.07
[500-1000)	**-18.84**	2.70	**-18.28**	17.06	**121.31**	144.87	**-6.06**	3.90	13.90	13.90
[1000-1500)	**-42.12**	2.66	**-48.35**	17.35	**79.07**	116.68	**-14.32**	3.90	13.41	13.41
[1500-2000]	**-67.59**	2.85	**-81.43**	17.34	**-24.98**	111.34	**-23.40**	3.85	13.41	13.41
Real Feature Costs & Asymmetric Misclassification Costs										
[0-500)	**7.33**	9.3	**22.74**	23.84	245.79	245.79	**-9.55**	5.84	17.7	17.7
[500-1000)	**-16.78**	2.66	**9.85**	13.31	**131.36**	143.3	**-47.61**	2.57	1.56	1.56
[1000-1500)	**-38.26**	3.04	**3.99**	11.7	**46.20**	114.23	**-84.79**	2.57	1.56	1.56
[1500-2000]	**-61.88**	2.97	**-2.54**	13.7	**-40.96**	107.14	**-125.69**	2.57	1.56	1.56
Synthetic Feature Costs & Symmetric Misclassification Costs										
[0-500)	307.39	307.39	418.34	418.34	723.36	723.36	231.93	231.93	298.01	298.01
[500-1000)	160.95	160.95	**245.75**	288.65	**579.25**	585.72	**63.59**	106.54	277.70	277.70
[1000-1500)	**60.30**	79.76	**163.80**	224.09	**444.42**	539.88	96.39	96.39	257.40	257.40
[1500-2000)	**31.86**	53.97	**138.78**	163.45	**378.43**	490.23	88.14	88.14	237.09	237.09
[2000-2500)	**10.43**	53.01	**108.69**	163.78	**364.03**	482.24	79.88	79.88	216.79	216.79
[2500-3000)	**-14.60**	62.66	**78.90**	164.75	**268.00**	458.89	71.63	71.63	196.48	196.48
[3000-3500)	**-39.64**	59.96	**48.83**	172.76	**171.91**	422.06	63.38	68.67	176.18	176.18
[3500-4000]	**-67.18**	63.68	**15.75**	172.13	**109.91**	412.11	**54.30**	63.66	153.84	153.84
Synthetic Feature Costs & Asymmetric Misclassification Costs										
[0-500)	306.19	306.19	441.29	441.29	728.57	728.57	219.04	219.04	276.32	276.32
[500-1000)	156.78	156.78	341.28	341.28	**599.02**	603.12	**88.34**	91.53	213.60	213.60
[1000-1500)	**66.57**	79.79	**260.09**	261.21	**505.80**	517.37	**-16.86**	49.39	162.52	162.52
[1500-2000)	**37.47**	60.62	**201.19**	204.31	**420.56**	519.29	**-54.31**	61.90	113.82	113.82
[2000-2500)	**14.84**	55.70	**161.24**	164.17	**320.32**	512.75	**-64.75**	61.90	**65.12**	68.39
[2500-3000)	**-9.19**	58.85	**144.24**	151.22	**211.26**	500.75	**-101.93**	61.90	**28.72**	34.73
[3000-3500)	**-33.22**	59.33	**132.84**	139.54	**248.73**	400.16	**-139.11**	61.90	**0.78**	14.67
[3500-4000]	**-59.66**	63.13	**126.43**	136.51	**206.06**	389.32	**-180.01**	61.90	**-18.10**	9.50

and it never looses. Finally, `Greedy` policy prematurely stops acquisition, having negative savings with respect to the `Markov Blanket` strategy.

3.5 Conclusion

The typical approach to feature acquisition has been greedy in the past primarily due to the sheer size of the possible subsets of features. We described a

general technique that can optimally prune the search space by exploiting the conditional independence relationships between the features and the class variable. We empirically showed that exploiting the conditional independence relationships can substantially reduce the number of possible subsets. We also introduced a novel data structure called Value of Information Lattice (`VOILA`) that can both efficiently reduce the search space using the conditional independence relationships and also can share probabilistic inference computations between different subsets of features. By using `VOILA`, we are able to add a look-ahead capability to the greedy acquisition policy, which would not be practical otherwise. We experimentally show on five real-world medical datasets that the greedy strategy often stops feature acquisition prematurely, performing worse than even a policy that acquires all the features.

Chapter 4

Label Acquisition During Inference For Relational Data

In this chapter, I discuss label acquisition during inference for relational data. Specifically, I assume that we are given an already trained classification model of a relational domain and a budget determining how many labels can be acquired. Our objective is then to determine the right set of labels to acquire during inference so that the classification performance on the remaining ones is maximized.

4.1 Introduction

Information diffusion, viral marketing, graph-based semi-supervised learning, and collective classification all attempt to exploit relationships in a network to reason and make inferences about the labels of the nodes in the network. The common intuition is that knowing (or inferring) something about the label of a particular node can tell us something useful about the other nodes' labels in the network. For instance, the labels of the linked nodes often tend to be correlated (not necessarily a positive correlation) for many domains; hence, finding the correct label of a node is useful for not only that particular node, but the inferred label also has an impact on the predictions that are made about the nodes in the rest of the network. Thus, it has been shown that methods such as collective classification, i.e., classifying the nodes of a network simultaneously, can significantly outperform content-only

classification methods, which make use of only the attributes of nodes and ignore the relationships between them (Chakrabarti et al., 1998; Neville and Jensen, 2000; Getoor et al., 2001; Taskar et al., 2002; Lu and Getoor, 2003a; Jensen et al., 2004; Macskassy and Provost, 2007; Sen et al., 2008). However, sometimes, the advantage of exploiting the relationships can become a disadvantage. In addition to the typical errors made by content-only classification models (errors due to model limitations, noise in the data, etc.), collective classification models can also make mistakes by propagating misclassifications in the network. This can sometimes even have a domino effect leading to misclassification of most of the nodes in the network. For example, consider a simple binary classification problem where an island of nodes that should be labeled with the positive label are surrounded with a sea of negatively labeled nodes. The island may be *flooded* with the labels of the neighboring sea of negative nodes.

This flooding of the whole network (or part of it) can occur for simple models such as iterative classification (Lu and Getoor, 2003a; Neville and Jensen, 2000) and label propagation (Zhu and Ghahramani, 2002). A misclassification can be propagated to the rest of the network, especially if the misclassification is systematic and common, such as misclassifying nodes as belonging to the majority class. Flooding can also happen for more complex models that define a global objective function to be optimized. For example, for pairwise Markov random field models (Taskar et al., 2002) with parameter values that prefer intra-class interactions over inter-class interactions, the most probable configuration of the labels might be the one where most of the network is labeled with one class. Or, for a graph mincut formulation

(Blum and Chawla, 2001), where we pay a penalty for inter-class interactions, the best objective value might be achieved by assigning only one label to each connected component in the graph.

One strategy for avoiding flooding is to have an expert in the loop during inference, who can guide the inference and constrain the solution space in the right directions by providing the correct labels for a few nodes. Depending on the application, labels can be acquired by asking the expert to rate specific items, a company can provide free samples to a small set of customers and customers' viral networking or purchasing behavior can be observed, or targeted laboratory experiments can be performed to determine protein functions, etc. However, providing these additional labels is often costly and we are often limited to operate within a given budget. As we show later, determining the optimal set of labels to acquire is intractable under relatively general assumptions. Therefore, we are forced to resort to approximate and heuristic techniques to get practical solutions.

In this chapter, we describe three polynomial-time label acquisition strategies. The first and most direct approach is based on approximating the objective function (which we define formally in Section 4.2) and greedily acquiring the label that provides the greatest improvement in the objective value. The second approach draws on an analogy between viral marketing and label acquisition, and translates one of the existing viral marketing formulations into a label acquisition strategy. The third approach, which we refer to as *reflect and correct*, is a simple yet effective acquisition method that learns the cases when a given collective classification model makes mistakes, finds islands of nodes that the collective model is likely to misclassify, and

suggests acquisitions to correct these potential mistakes (Bilgic and Getoor, 2008, 2009, 2010).

In addition to these three methods, we also experiment with acquisition strategies that are based on network structural measures such as node degree and network clustering. To compare the different acquisition strategies, we use two representative collective models: one that consists of a collection of local classifiers, and one that defines and optimizes a global objective function. Using synthetic datasets, we analyze the cases when flooding might happen and its degree of severity. We compare the acquisition strategies on the synthetic datasets under varying settings and on real-world datasets, and we empirically show that the *reflect and correct* method we propose significantly outperforms all of the other methods.

The label acquisition problem has received ample attention within the context of active learning (Cohn et al., 1996; McCallum and Nigam, 1998; Tong and Koller, 2001). There are two main differences between the scenario we address and the active learning scenario. First, active learning has traditionally been concerned with non-relational data; here, we are interested in network data. The second (and the biggest) difference is that we assume that we have available an already trained model of the domain, and thus the learning has been done offline, but we have the option to acquire labels to seed the classification during inference. This is the setting Rattigan et al. (2007b) introduced and referred to as "active inference." They looked at the relational network classifier, introduced by Macskassy and Provost (2003), in which there are no node attributes; only labels are propagated. Here, we look at networks

in which the nodes have attribute information and compare to the structural strategy that they introduced.

This chapter is organized as follows: we formulate the label acquisition problem and state the objective function in Section 4.2. Then, we explain the three approaches in Sections 4.3.1, 4.3.2, and 4.3.3. We then show experimental results on both synthetic and real datasets in Section 4.4. Finally, we discuss summarize the contributions in Section 4.5 and then conclude in Section 4.6.

4.2 Problem Formulation

We begin by reviewing the collective classification problem and define the objective function for label acquisition for collective classification. In this problem, we assume that our data is represented as a graph with nodes and edges, $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. Each node $V_i \in \mathbf{V}$ is described by an attribute vector $\vec{X}_i$ and a class label Y_i pair, $V_i = \langle \vec{X}_i, Y_i \rangle$. $\vec{X}_i$ is a vector of individual attributes $\langle X_{i1}, X_{i2}, \ldots, X_{ip} \rangle$. The domain of X_{ij} can be either discrete or continuous whereas the domain of the class label Y_i is discrete and denoted as $\{y_1, y_2, \ldots, y_m\}$. Each edge $E_{ij} \in \mathbf{E}$ describes some sort of relationship between its endpoints, $E_{ij} = \langle V_i, V_j \rangle$. Examples include:

social networks: Here, the nodes are people, the attributes may include demographic information such as age and income and the edges are friendships. The labels indicate categories of people, for example we may be interested in labeling the people that are likely to partake in some activity (e.g., smoking,

IV drug use), have some disease (e.g., tuberculosis, obesity), or exhibit some behavior (buying a product, spreading a rumor).

citation networks: the nodes are publications, the attributes include content information and the edges represent citations. The labels may be the topics of the publications, or an indication of the reputation of the paper, for example whether the paper is seminal or not.

biological networks: where for example, the nodes represent proteins, attributes include annotations, and edges represent interactions. In this domain for example, we may be interested in inferring protein function.

4.2.1 Collective Classification

In graph data, the labels of neighboring nodes are often correlated (though not necessarily positively correlated). For example, friends tend to have similar smoking behaviors, papers are likely to have similar topics to the papers that they cite, and proteins are likely to have complementary functions. Exploiting these correlations can significantly improve classification performance over using only the attributes, $\vec{X}_i$, for the nodes. However, when predicting the label of a node, the labels of the related instances are also unknown and need to be predicted. *Collective classification* is the term used for simultaneously predicting the labels $\mathbf{Y}$ of $\mathbf{V}$ in the graph $\mathbf{G}$, where $\mathbf{Y}$ denotes the set of labels of all of the nodes, $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_n\}$. In general, the label Y_i of a node can be influenced by its own attributes $\vec{X}_i$ as well as the labels Y_j and attributes $\vec{X}_j$ of other nodes in the graph.

There are many collective classification models proposed to date that make different modeling assumptions about these dependencies. They can be grouped into two broad categories. In the first category, *local collective classification models*, the collective models consist of a collection of local vector-based classifiers, such as logistic regression. For the this category of collective models, each object is described as a vector of its local attributes $\vec{X}_i$ and an aggregation of attributes and labels of its neighbors. Examples include Chakrabarti et al. (1998), Neville and Jensen (2000), Lu and Getoor (2003a), Macskassy and Provost (2007), and McDowell et al. (2007). The second category of collective classification models are *global collective classification models*. In this case, the collective classification is defined as a global objective function to be optimized. In many cases, a relational graphical model is learned over all the attributes and labels in the graph, and a joint probability distribution over these attributes and labels is learned and optimized. Examples of this category include conditional random fields (Lafferty et al., 2001), relational Markov networks (Taskar et al., 2002), probabilistic relational models (Getoor et al., 2002), and Markov logic networks (Richardson and Domingos, 2006).

In this chapter, we use an example model from each category, which we explain briefly here. For the local collective classification model, we use Iterative Classification Algorithm (`ICA`) (Neville and Jensen, 2000; Lu and Getoor, 2003a), and, for the global collective classification model, we use a pairwise Markov Random Fields (`MRF`) based on the relational Markov network of Taskar et al. (2002). We first introduce notations and assumptions common to both models and then describe the two approaches.

Let $\mathbf{N}_i$ denote the labels of the neighboring nodes of V_i, $\mathbf{N}_i = \{Y_j \mid \langle V_i, V_j \rangle \in \mathbf{E}\}$. A general assumption that is made is the Markov assumption that Y_i is directly influenced only by $\vec{X}_i$ and $\mathbf{N}_i$. Given the values of $\mathbf{N}_i$, Y_i is independent of $\mathbf{Y} \setminus \mathbf{N}_i$ and is independent of $\mathbf{X} \setminus \{\vec{X}_i\}$, where $\mathbf{X}$ denotes the set of all attribute vectors in the graph, $\mathbf{X} = \{\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_n\}$. That is, once we know the the values of $\mathbf{N}_i$, then Y_i is independent of attribute vectors $\vec{X}_j$ of all neighbors and non-neighbors, and it is independent of labels Y_j of all non-neighbors.

4.2.1.1 Iterative Classification Algorithm (`ICA`)

In the `ICA` model, each node in the graph is represented as a vector that is a combination of node features, $\vec{X}_i$, and features that are constructed using the labels of the nodes' immediate neighbors. Because each node can have a varying number of neighbors, we use an aggregation function over the neighbor labels in order to get a fixed-length vector representation. For example, the `count` aggregation constructs a fixed-size feature vector by counting how many of the neighbors belong to each label; other examples of aggregations include `proportion`, `mode`, etc. Once the features are constructed, then an off-the-shelf probabilistic classifier can be used to learn $P(Y_i \mid \vec{X}_i, \texttt{aggr}(\mathbf{N}_i))$, where `aggr` is an aggregate function that converts a set of inputs into a fixed length vector. One can use a single classifier to learn $P(Y_i \mid \vec{X}_i, \texttt{aggr}(\mathbf{N}_i))$ or can use a structured classifier to learn $P(Y_i \mid \vec{X}_i)$ and $P(Y_i \mid \texttt{aggr}(\mathbf{N}_i))$ separately, which can be combined in a variety of ways to compute $P(Y_i \mid \vec{X}_i, \texttt{aggr}(\mathbf{N}_i))$.

A key component of this approach is that during inference, the labels of the neighboring instances are often not known. ICA addresses this issue, and performs collective classification, by using the predicted labels for the neighbors for feature construction. ICA iterates over all nodes making a new prediction based on the predictions made for the unknown labels of the neighbors in the previous iteration; in the first step of the algorithm, initial labels can be inferred based solely on attribute information, or based on attribute and any observed neighboring labels. ICA learning is typically done using fully labeled training data, however there are also approaches that use semi-supervised techniques for learning ICA (Lu and Getoor, 2003b; Xiang and Neville, 2008).

4.2.1.2 Pairwise Markov Random Fields (MRF)

Now, we briefly describe the MRF model we used. In an MRF, the joint probability of $P(\mathbf{Y} \mid \mathbf{X})$ is given by:

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z} \prod_{Y_i \in \mathbf{Y}} \phi(Y_i, \vec{X}_i) \phi(Y_i, \mathbf{N}_i)$$

where ϕ are the "compatibility" functions that in effect capture the degree and the strength of the relationships between different values of Y_i and $\vec{X}_i$ and Y_i and $\mathbf{N}_i$, and Z is the normalization function that ensures $P(\mathbf{Y} \mid \mathbf{X})$ is a legitimate probability distribution. For example, in social networks, these compatibility functions can be considered to capture the degree of correlations between the smoking behaviors of friends in a network.

Note that in this representation, the number of arguments for ϕ are neither fixed nor uniform; that is, for one particular node $\phi(Y_i, \mathbf{N}_i)$ can have just two arguments whereas for a different node it can have hundreds of arguments. This property is undesirable for many reasons including representational inefficiency, lack of sufficient data for accurate parameter estimation, and difficulty of generalizing from train data to the test data. To get around these problems, one trick is to assume a functional form for the ϕ. For example, we can approximate the interactions between many nodes as the product of the pairwise interactions; this assumption leads to pairwise Markov Random Fields:

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z} \prod_{Y_i \in \mathbf{Y}} \left(\prod_{j=1}^{p} \phi(Y_i, X_{ij}) \right) \left(\prod_{Y_j \in \mathbf{N}_i} \phi(Y_i, Y_j) \right) \tag{4.1}$$

One other trick to make sure that the ϕ are generalizable from train to test data and also to make sure that $P(\mathbf{Y} \mid \mathbf{X})$ is integrable is to represent ϕ as log-linear combinations of a set of indicator functions of the form $f_{y_i,x_{ij}}(Y_i, X_{ij}) \triangleq \sigma(Y_i = y_i, X_{ij} = x_{ij})$, and $f_{y_i,y_j}(Y_i, Y_j) \triangleq \sigma(Y_i = y_i, Y_j = y_j)$. Then, the compatibility functions are represented as:

$$\phi(Y_i, X_{ij}) = e^{\left(\sum_{ij} w_{y_i,x_{ij}} f_{y_i,x_{ij}}(Y_i,X_{ij}) \right)}; \phi(Y_i, Y_j) = e^{\left(\sum_{ij} w_{y_i,y_j} f_{y_i,y_j}(Y_i,Y_j) \right)}$$

where ws are weights to be learned from the train data. With this representation, the products in Equation (4.1) turn into sums in the exponent of e. Then, maximum

likelihood learning can be done by the taking log of $P(\mathbf{Y} \mid \mathbf{X})$, which is now sum of the products of the weights and the indicator features, and maximizing it through gradient ascent methods (Taskar et al., 2002).

4.2.2 Label Acquisition

For active inference for both `ICA` and `MRF`, we assume that we are given a training graph $\mathbf{G}^{tr}(\mathbf{V}^{tr}, \mathbf{E}^{tr})$ where labels of all the nodes are known. Let `CM` represent the collective model we use, here either `ICA` or `MRF`. We train our collective model `CM` using this training graph. Given a test graph $\mathbf{G}$, a trained model `CM` and assuming the values of the attribute vectors $\mathbf{X}$ are known, but the labels for the nodes are unknown, our goal is to correctly predict $\mathbf{Y}$. We assume we are given a cost for misclassifying a node; when we classify a node as y_k whereas the correct assignment is y_l, we incur a cost of c_{kl}. The expected misclassification cost ($EMC_{\texttt{CM}}$) for a node when using the collective model `CM` is then given by:

$$EMC_{\texttt{CM}}(Y_i \mid \mathbf{X} = \mathbf{x}) = \min_{y_k} \sum_{y_l \neq y_k} P_{\texttt{CM}}(Y_i = y_l \mid \mathbf{X} = \mathbf{x}) \times c_{kl}$$

The total expected misclassification cost is then sum of the expected misclassification costs for the individual nodes:

$$\sum_{Y_i \in \mathbf{Y}} EMC_{\texttt{CM}}(Y_i \mid \mathbf{X} = \mathbf{x}).$$

As mentioned in the introduction, we are interested in settings where we are able acquire additional information, or to ask for the labels for some of the nodes. More formally, we consider the case where we can acquire the values for a subset of the labels $\mathbf{A} \subseteq \mathbf{Y}$. The acquisition of $\mathbf{A}$ changes the misclassification cost as follows:

$$\sum_{Y_i \in \mathbf{Y} \setminus \mathbf{A}} EMC_{\texttt{CM}}(Y_i \mid \mathbf{X} = \mathbf{x}, \mathbf{A})$$

However, we do not know the values of the labels in $\mathbf{A}$ before we acquire them. Thus, we take an expectation over possible values.

$$\sum_{Y_i \in \mathbf{Y} \setminus \mathbf{A}} \sum_{\mathbf{a}} P(\mathbf{A} = \mathbf{a}) EMC_{\texttt{CM}}(Y_i \mid \mathbf{X} = \mathbf{x}, \mathbf{A} = \mathbf{a})$$

In this general setting, we also attach costs to acquiring labels. Let the cost of acquiring the value of the label Y_i be C_i. Extending it to sets, $C(\mathbf{A}) = \sum_{Y_i \in \mathbf{A}} C_i$. Then, the total cost we incur is just the sum of the acquisition cost and the expected misclassification cost:

$$L(\mathbf{A}) = C(\mathbf{A}) + \sum_{Y_i \in \mathbf{Y} \setminus \mathbf{A}} \sum_{\mathbf{a}} P(\mathbf{A} = \mathbf{a}) EMC_{\texttt{CM}}(Y_i \mid \mathbf{X} = \mathbf{x}, \mathbf{A} = \mathbf{a}) \tag{4.2}$$

Given a spending budget B, the label acquisition problem, and our objective, is then to find the optimal subset

$$\mathbf{A}^* = \operatorname*{argmin}_{\mathbf{A} \subseteq \mathbf{Y}, C(\mathbf{A}) \leq B} L(\mathbf{A})$$

minimizing the sum of expected misclassification cost and acquisition cost.

Finding the optimal $\mathbf{A}^*$ requires us to evaluate the objective function for each candidate $\mathbf{A}$ along with an efficient search and exploration of the candidate space. Krause and Guestrin (2005a) discuss this problem in the context of value of information calculations in graphical models. They associate a reward for observing the value at a node, which is equivalent to acquiring a label in our case, and they show that reward computations are $\#\mathbf{P}$-complete even for discrete polytrees. They also show that finding the optimal set to acquire in batch mode, where the acquisition decisions are made all at once, is $\mathbf{NP}^{\mathbf{PP}}$-complete for discrete polytrees. Finally, they show that finding the optimal set in a conditional plan, that is the next acquisition is conditioned on the previous acquisitions, is $\mathbf{NP}^{\mathbf{PP}}$-hard for discrete polytrees. Given that we are considering arbitrary networks, such as citation, friendship, and protein networks, finding the optimal solution is at least as hard as, if not harder than, considering discrete polytrees. The details of these theoretical limits can be found in (Krause and Guestrin, 2005a).

4.3 Active Inference

Since finding the optimal solution to the label acquisition problem is intractable under relatively general assumptions, we must resort to approximate and/or heuristic acquisition techniques. In this chapter, we introduce three such techniques. Each technique associates a *utility* value with each label (or sets of labels) and makes acquisition decisions based on the utility values. The first strategy that we propose

approximates the objective function and defines the utility of a label in terms of the improvement it achieves in the objective value. The second method draws an analogy between viral marketing and active inference and associates utilities with labels accordingly. The third is a simple yet effective and intuitive approach based on learning and predicting the misclassifications of a collective classifier.

4.3.1 Approximate Inference and Greedy Acquisition (`AIGA`)

There are two reasons why finding the optimal set $\mathbf{A}*$ is intractable: 1) unless the probability distribution for $\mathbf{A}*$ can be factored with the acquisition of a single label Y_i, we need to consider all possible subsets $\mathbf{A} \subseteq \mathbf{Y}$, which is exponential in the size of $\mathbf{Y}$, and 2) for each candidate set $\mathbf{A}$, we need to compute the value of the objective function $L(\mathbf{A})$ (Equation (4.2.2)), which requires us to compute exact probability distributions over $\mathbf{Y}$.

To tackle these two obstacles, we first introduce the most obvious approach: approximate inference and greedy acquisition (`AIGA`). In `AIGA`, instead of considering all candidate sets, we consider acquiring one label at a time. That is, we define the *utility* of a label to be the amount of improvement it provides in the current objective value and we greedily acquire the label that has the highest *utility*:

$$utility_{aiga}(Y_i) \triangleq L(\mathbf{A} \cup \{Y_i\}) - L(\mathbf{A})$$

In essence, the $utility_{aiga}$ function is computing the *expected value of information* for each label (Howard, 1966).

To address the intractability of the exact probability computations, we resort to approximate inference techniques. For the collective models (such as `ICA`) that are a collection of local classifiers, we use iterative approaches to approximate the conditional probability distributions for the labels. For the collective models that define and optimize a global objective function (such as `MRF`), there exist a variety of approximate inference techniques, including loopy belief propagation (Yedidia et al., 2000), variational methods (Jordan et al., 1999), and Gibbs sampling (Gilks et al., 1996); in this chapter, we use loopy belief propagation.

With these two approximations, `AIGA` iteratively finds the label that has the highest $utility_{aiga}$, adds it to the acquisition set, and repeats this step until the budget is exhausted. Note that, even though we make the problem tractable through approximate inference and greedy selection, we still need to run approximate inference for each iteration, for each node, and for each possible value of the label of the node under consideration. This requirement makes this approach still quite expensive, especially if the number of nodes is relatively high and the underlying approximate inference technique is slow. Additionally, the accuracy of this method depends heavily on the precision of the estimated probability values. If the probability estimates are not well-calibrated, then the expected misclassification costs will be incorrect (Zadrozny and Elkan, 2001), making the $utility_{aiga}$ values inaccurate.

4.3.2 Viral Marketing Acquisition (VMA)

Another approach to label acquisition is based on an analogy to viral marketing (Richardson and Domingos, 2002; Kempe et al., 2003; Leskovec et al., 2007a). In the viral marketing setting, we have customers that are potential buyers of a product, and the customers have relationships between each other, such as family, friendship, co-worker, etc. When a customer buys a product, the customer advertises it (by word of mouth) to his or her neighbors in the network. The objective of viral marketing is to maximize the sales for a product by marketing it to the right set of customers, while minimizing the marketing costs. Thus, similar to the label acquisition problem, there is then the question of which subset of customers we should target, in the hope that these customers will like the product, buy it, and recommend it to their neighbors, who will hopefully buy and recommend it in turn.

The analogous mapping to label acquisition for collective classification is as follows. There are nodes (customers) that we need to classify and we have the choice to acquire the labels for (market to) some of them. Our task is to acquire the labels for the right subset of nodes so that the number of correctly classified nodes (the customers who buy the product) in the end is maximized, while minimizing the acquisition cost. This analogy between viral marketing and label acquisition is summarized in Table 4.1.

There are many viral marketing approaches that differ in the formulation of the problem, the assumptions that they make, and the solutions that they offer (Richardson and Domingos, 2002; Kempe et al., 2003; Leskovec et al., 2007a). We

Table 4.1: The analogy between viral marketing and active inference.

	Viral Marketing	Active Inference
Objects	Customers	Nodes
States	Bought / Did not buy	Classified correctly / Misclassified
Action	Market a product to a subset of customers	Acquire labels for a subset of nodes
Objective	Maximize the number of customers that buy the product	Maximize the number nodes that are classified correctly
Constraint	A budget for the marketing costs	A budget for acquisition costs

chose the formulation of Richardson and Domingos (2002); an advantage of their approach is that it has an exact solution.

In this formulation, we introduce a new random variable T_i for each node V_i, which indicates whether Y_i is predicted correctly. Whether a prediction for a node is correct depends on the informativeness of the node's attributes X_i, whether its neighbors $\mathbf{N}_i$ are classified correctly, and which labels are acquired, $\mathbf{A}$. Following Richardson and Domingos (2002), we make the assumption that this probability is a linear combination of a local probability and a relational probability as follows:

$$P(T_i \mid \mathbf{N}_i, X_i, \mathbf{A}) \triangleq \beta_i P_l(T_i \mid X_i, \mathbf{A}) + (1 - \beta_i) P_r(T_i \mid \mathbf{N}_i, \mathbf{A})$$

where β_i denotes how much the label of a node depends on the node's local attributes versus its neighbors. Here, P_l stands for the local probability which is defined as:

$$P_l(T_i \mid X_i, \mathbf{A}) \triangleq \begin{cases} 1 & \text{if } Y_i \in \mathbf{A} \\ \max\limits_{y_k} P(Y_i = y_k \mid X_i) & \text{otherwise} \end{cases} \tag{4.3}$$

and P_r stands for the relational probability which is a linear combination of the statuses of the neighbors:

$$P_r(T_i \mid \mathbf{N}_i, \mathbf{A}) = \frac{1}{|\mathbf{N}_i|} \sum_{Y_j \in \mathcal{N}_i} T_j.$$

The probability $P(Y_i = y_k \mid X_i)$ in Equation (4.3) can be computed by learning a classifier on the nodes of the train graph $\mathbf{G}^{tr}$.

The objective here is to maximize the total probability of correctly classifying the nodes in the network. With this objective, we define the $utility_{vma}(Y_i)$ to be the increase in this total probability that Y_i causes once it is acquired. To compute $utility_{vma}(Y_i)$, we first calculate two intuitive measures. The first one corresponds to how much a unit change in $P_l(T_i \mid X_i, \mathbf{A})$ affects the total probability in the network:

$$\Delta(Y_i) \triangleq \sum_{V_j \in \mathbf{V}} \frac{\partial P(T_j = 1 \mid X_j, \mathbf{A})}{\partial P_l(T_i \mid X_i, \mathbf{A})}$$

The second one measures how much an instance's probability of correct classification is increased when we acquire the label for it:

$$\Delta P(Y_i) = \beta_i \left(P_l(T_i \mid X_i, \mathbf{A} \cup Y_i) - P_l(T_i \mid X_i, \mathbf{A}) \right)$$

Then, the effect that acquiring a label Y_i will have in the network, i.e., the $utility_{vma}$ of a label is, just a product of the two:

$$utility_{vma}(Y_i) \triangleq \Delta(Y_i)\Delta P(Y_i).$$

We omit some of the details about how to derive these equations. The interested reader can refer to Richardson and Domingos (2002).

With these assumptions, our formulation is the same as that of Richardson and Domingos (2002) with one subtle difference. In the viral marketing domain, when a person is marketed a product, there is still a non-zero probability of that person not buying the product. In label acquisition, however, we assume that we can acquire labels with perfect information; that is, there is no uncertainty about a node's label after we acquire it. Because this particular formulation of viral marketing has an exact solution, we compute the $utility_{vma}$ values for all candidate labels only once, and then we acquire the labels that have the highest utility values. This property of the approach makes it quite fast and thus attractive.

4.3.3 Reflect and Correct (`RAC`)

The next method that we introduce is based on a simple intuition: the sets of nodes that the collective classification model misclassifies tend to be clustered together because misclassifying one node makes it very likely that its neighbors will be misclassified as well (propagation of incorrect information). Thus, there are islands (or peninsulas) of misclassification in the graph – sets of connected nodes

that are misclassified. We call such nodes the *flooded* nodes. If we can find these islands of misclassification, then we can potentially trigger correct classification of those islands by acquiring labels for a few of the nodes in the islands. The question is then how to find the islands of misclassification.

We first focus on finding out when a prediction for a particular node is incorrect. We again associate a random variable T_i with each $V_i \in \mathbf{V}$, like we did in the viral marketing formulation, but this time we take a reverse perspective and T_i denotes whether the prediction for Y_i was indeed incorrect. Additionally, instead of using a viral marketing approach to learn and predict T_i, we formulate it as a classification problem by constructing features that are possible indicators of whether a node is misclassified, and learn a classifier to capture the dependence of T_i on the constructed features. Then, the acquisition problem can be solved by running the collective inference on the test graph, predicting which nodes are misclassified, acquiring a label for a central node among the potentially flooded ones, and repeating the process until the budget is exhausted. This process is illustrated in Figure 4.1. Because we reflect back on our inference results on the test graph and try to correct the mistakes by acquiring a label, we call this method *reflect and correct* (`RAC`).

Many different kinds of features can be constructed to be used for predicting whether a node is misclassified. In this chapter, we present the general framework for `RAC` and construct and experiment with only three simple and intuitive features as examples. The list can be extended with more features; especially, one can think of incorporating domain knowledge as features as well. The first of the three features we constructed is based on the content information of the node, the second one is

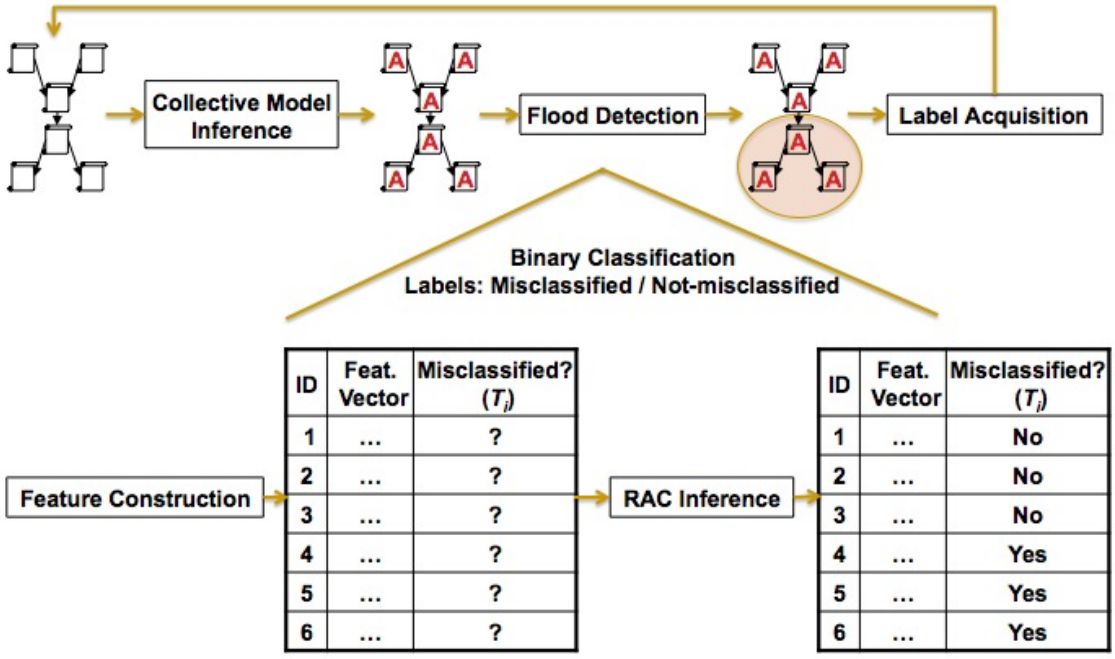


Figure 4.1: Active inference using the RAC method. We iteratively label the nodes using the collective model, predict which nodes are misclassified, acquire the central node among the misclassified ones, and repeat the process until the budget is exhausted. To predict which nodes are misclassified, we use a classifier whose input consists of a set features that are constructed using the content information of the nodes, information from the neighbors, and global statistics.

based on the neighbors of the node, and the last one is based on global statistics. Intuitively, the content indicator captures how much the node attributes disagree with the classification decisions of the collective model. The relational indicator captures how likely it is that the neighbors of a node are also misclassified. Lastly, the global indicator captures how different the posterior distribution of the labels is from the expected prior distribution. We next explain these features in detail and provide their formal definitions.

The *content indicator* measures how far the prediction of the collective model is from the truth according to the attributes. Assume that the collective model predicts $Y_i = y_j$. Then, we define the content indicator for node V_i to be:

$$ci_i \triangleq 1 - P(Y_i = y_j \mid X_i)$$

Again, we can compute $P(Y_i = y_j \mid X_i)$ by learning a local classifier for the nodes of the train graph $\mathbf{G}^{tr}$. The intuition behind the content indicator ci is that if the attributes of a node disagree with the prediction made by the collective model, then it is a signal for a possible misclassification. The content indicator is a measure of the strength of the disagreement between the local classifier and the collective model. However, the content indicator alone will not be sufficient for misclassification detection; otherwise, we could just replace the collective model with the local classifier.

The *relational indicator* captures how likely it is that a node's neighbors are also misclassified. The intuition is that if a node's neighbors are misclassified, then the node itself is probably misclassified as well (because the classification model is a collective one). There are different possibilities for defining the relational indicator; for instance, it can be defined as a recursive function of T_i, and then it can be computed iteratively. We take the simplest approach and define it as the average of the content indicators, ci_j, of the neighbors of the node V_i.

$$ri_i \triangleq \frac{1}{|\mathcal{N}_i|} \sum_{Y_j \in \mathcal{N}_i} ci_j$$

Finally, the *global indicator* captures the difference between our prior belief about the class distributions and the posterior distribution that we get based on the predictions. For example, based on our prior belief, if we expect to classify 20%

of the nodes with label y_j, but the collective model predicts 60% of the nodes as label y_j, then some of the nodes that are classified as y_j are probably misclassified. Let the prior distribution of the class y_j be denoted by $Prior(y_j)$ and let the posterior distribution based on the predictions of the collective model be denoted by $Posterior(y_j)$. Then, we define the global indicator for the node V_i that is predicted as y_j as follows:

$$gi_i \triangleq \frac{Posterior(y_j) - Prior(y_j)}{1 - Prior(y_j)}$$

Having constructed these three features, we learn a classifier for estimating the distribution $P(T_i \mid ci_i, ri_i, gi_i)$. To learn this classifier, we need training data, which requires four pieces of information per node: the three indicators described above, and the value of T_i. To obtain this information, we use our collective model and the training graph $\mathbf{G}^{tr}$. As a first step, we run collective inference on $\mathbf{G}^{tr}$ assuming the labels are unknown, to obtain a new graph where the node labels are now the predicted ones. Let this new graph be called the prediction graph $\mathbf{G}^{pr}$. To obtain the content indicator ci_i, we first learn a content-only classifier on the attribute vectors of the nodes of the training graph $\mathbf{G}^{tr}$ and then use this content-only classifier and the predicted labels in the prediction graph $\mathbf{G}^{pr}$ to compute ci_i. To obtain the relational indicators ri_i, we use the content indicators that we just computed. To obtain the global indicators gi_i, we collect the prior class distribution statistics on the training graph $\mathbf{G}^{tr}$ and the posterior class distribution statistics on the prediction graph $\mathbf{G}^{pr}$. Finally, to obtain the value of T_i, we compare the correct

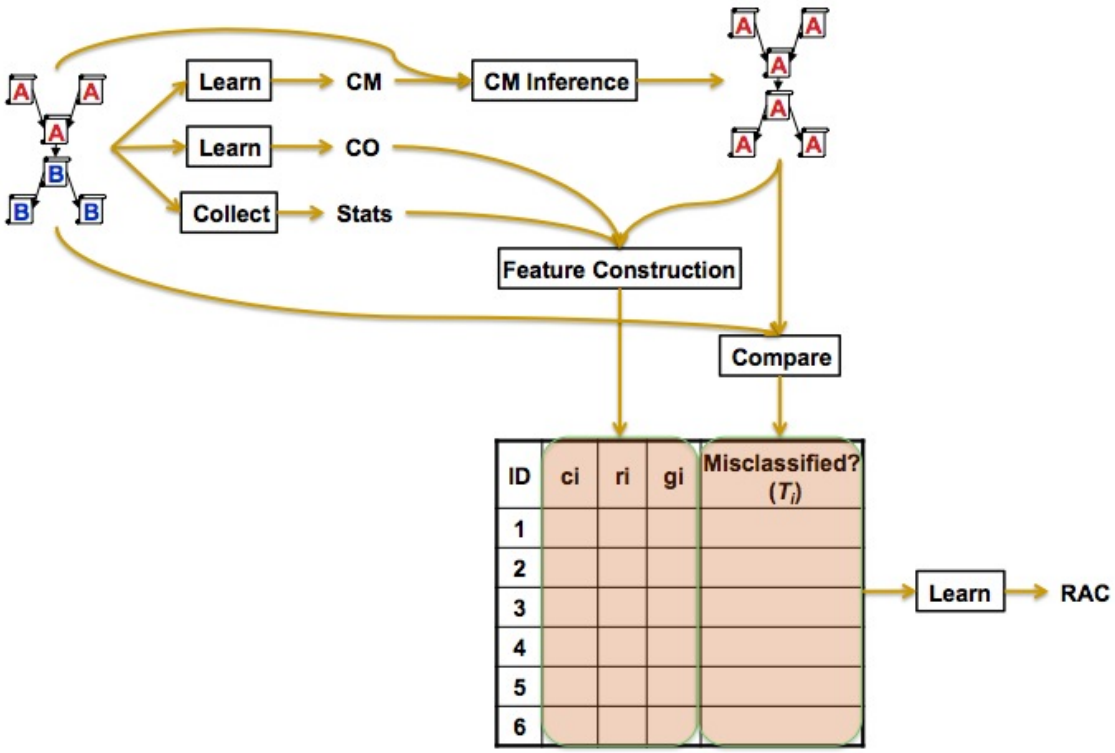


Figure 4.2: The process of learning $P(T_i \mid ci_i, ri_i, gi_i)$. We the collective model `CM` to predict the labels for the training graph. To construct the content indicator, we use the predicted labels and a content-only classifier that was learned on the training graph. To construct the global indicator, we collect prior class distribution statistics on the training graph and the posterior class distribution statistics on the predicted labels. To obtain the class information T_i, we compare the true labels and the predicted labels.

labels from the training graph $\mathbf{G}^{tr}$ with the predicted labels from the prediction graph $\mathbf{G}^{pr}$. Having constructed the training data, we can use any probabilistic classifier to learn the distribution $P(T_i \mid ci_i, ri_i, gi_i)$. This process is illustrated in Figure 4.2.

The question that remains to be answered is how to define the utility of label Y_i given $P(T_i \mid ci_i, ri_i, gi_i)$. The most obvious way is to have $utility_{rac}Y_i \triangleq P(T_i \mid ci_i, ri_i, gi_i)$. However, given that we have a limited budget, we want each of the acquisitions to correct as many misclassifications as possible. The node that has the

highest probability of misclassification $P(T_i \mid ci_i, ri_i, gi_i)$ can be an isolated node in the network; then acquiring the label for that node might not have a big impact on the predictions for the labels of the rest of the nodes. Based on these intuitions, we want the utility of a label to be a function of whether the corresponding node is misclassified, and how many misclassified neighbors it has. More formally:

$$utility_{rac}(Y_i) \triangleq \begin{cases} 0 & \text{if } P(T_i \mid ci_i, ri_i, gi_i) < \sigma \\ 1 + \sum_{Y_j \in \mathbf{N}_i} \delta(P(T_j \mid ci_j, ri_j, gi_j) \geq \sigma) & \text{otherwise} \end{cases}$$

where $\delta(predicate) = 1$ if the *predicate* is true, 0 otherwise, and σ is the threshold used to decide if a node is misclassified.

We can have a fixed threshold σ, like 0.5, however, we want to set it adaptively based on the training data and the underlying collective model. Specifically, we want σ to be a function of the prior probability of the misclassifications of the collective model on the training data. The way we set it is as follows: let p be the percentage of the nodes in the training data that are misclassified by the collective model. We first sort all the test data in decreasing order of $P(T_i \mid ci_i, ri_i, gi_i)$ and then we set σ to the misclassification probability of the last node in the top p percent in this sorted list. Instead of continuously updating the σ after each acquisition step, we fix it at this value before any acquisitions are made because we expect the percentage of misclassified nodes in the test graph to decrease as we acquire more labels. With

the misclassification predictor learned and the utility function defined, the missing pieces of the acquisition process, which is illustrated in Figure 4.1, are now complete.

RAC is a general active inference framework that can be applied using varios features and utility functions; we have discussed three example features and an example utility function above. The relative merits of each feature and utility function depend on the domain, the noise level in the attributes of the nodes, the strength of the correlation between the node labels, and the degree of class skew present in the data. For example, the benefit of the content indicator correlates negatively with the noise level in the attributes of the nodes; the relational indicator's benefit correlates with the degree of label correlation; the global indicator's benefit correlates with the degree of class skew in the data. Similarly, a utility function that takes the misclassification predictions for neighbors into account is useful for correcting flooded regions, whereas the utility function that takes only the individual scores into account is useful for domains where flooding is not the main problem but instead the focus is to correct single mistakes.

4.3.4 Generalized Utility-based Active Inference

In this subsection, we describe a generic active inference algorithm that unifies the three acquisition methods described above. The algorithm also serves as a generic utility-based active inference technique. In this general algorithm, which we formally describe in Algorithm 4.1, we iteratively find the label Y_i that has the highest utility and whose acquisition cost does not cause us to exceed the given

budget, we add it to our acquisition set, and then repeat the process until we exhaust the budget.

Algorithm 4.1: Generalized utility-based active inference algorithm.

Input: $\mathbf{G}$ – the test graph, $\texttt{CM}$ – the learned collective model, c_{ij} – misclassification costs, C_i – the acquisition costs, B – the budget
Output: $\mathbf{A}$ – the set of acquisitions

1 $\mathbf{A} \leftarrow \emptyset$
2 **while** $C(\mathbf{A}) < B$
3 $\quad Y_{max} \leftarrow nil$
4 $\quad maxValue \leftarrow -\infty$
5 $\quad$ **for** $Y_i \in \mathbf{Y} \setminus \mathbf{A}$
6 $\quad\quad utility_{Y_i} \leftarrow utility(Y_i \mid \mathbf{X}, \mathbf{A}, \mathbf{Y} \setminus \mathbf{A}, c_{ij}, \texttt{CM})$
7 $\quad\quad$ **if** $utility_{Y_i} > maxValue \wedge C(\mathbf{A} \cup \{Y_i\}) \leq B$ **then**
8 $\quad\quad\quad maxValue \leftarrow utility_{Y_i}$
9 $\quad\quad\quad Y_{max} \leftarrow Y_i$
10 $\quad \mathbf{A} \leftarrow \mathbf{A} \cup \{Y_{max}\}$

The utility function at step six of the algorithm is replaced with $utility_{aiga}$, $utility_{vma}$, and $utility_{rac}$ for the acquisitions we described previously. The utility function in its most general form is a function of the label under consideration Y_i, the set of all attribute vectors $\mathbf{X}$, what has been acquired thus far $\mathbf{A}$, the set of remaining labels $\mathbf{Y} \setminus \mathbf{A}$, the cost model c_{ij}, and the underlying collective model, $\texttt{CM}$.

This algorithm is very similar to the general utility-based active learning algorithms used by many different techniques such as Lewis and Gale (1994), Melville and Mooney (2004), Roy and McCallum (2001), and Saar-Tsechansky and Provost (2004) with one notable difference; in active learning, we update the underlying classification model at each step. However, in active inference, we assume that we have enough training data to learn the collective model. One thing that might not be obvious from this algorithm is that the utility computation at step six of

the algorithm requires running the collective inference for some of the acquisition strategies.

4.4 Experimental Evaluation

We begin our experimental evaluation with a study aimed at better understanding the effects of flooding in collective classification. Then, we evaluate our proposed label acquisition strategies against a variety of baselines. We evaluate on both synthetic and real-data and perform a relatively comprehensive exploration of options and settings for the algorithms.

4.4.1 Understanding Flooding

Collective classification models classify a node in a network based on both local attributes and characteristics of the node, such as words in a document, and information contained in the neighboring nodes, such as topics of the documents that reference this document. Thus, a prediction for a node in a given network both affects and depends on the predictions for the other nodes in the network. Due to these structural dependencies, in addition to the typical errors made by the non-collective models (errors due to noise in the data, incorrect modeling assumptions, etc.), collective models can make a second type of error by spreading an incorrect decision to the neighboring nodes or by committing to an incorrect decision jointly. As mentioned in the introduction, we call this kind of error *flooding*. Depending on the characteristics of the data and the underlying collective model, flooding can be

quite severe; in extreme cases, most of the network can be flooded with just one label.

Before we evaluate different acquisition strategies, we explore the spectrum of flooding as a function of noise in the attributes and correlation of the node labels in the network. In order to quantify flooding, we introduce two measures. The first measure, which we refer to as *perfect information* (PI) accuracy, is the accuracy that corresponds to the setting where to classify a node, the collective model is allowed to look at the true labels (according to the ground-truth) of the node's neighbors instead of the predicted labels. Obviously, this is a hypothetical situation, but it is quite useful for quantifying floods and comparing different collective models as to how well their modeling assumptions fit to the data and how well they can exploit attribute and neighborhood information. The second measure, which we refer to a *no acquisition* (NOACQ) accuracy, is the accuracy that is achieved before any label is acquired. Given these two measures, we define the *flood percentage* (FP) as the difference between the two:

$$\mathtt{FP} \triangleq \mathtt{PI} - \mathtt{NOACQ}$$

To explore the spectrum of flooding, we generated synthetic data where we change the attribute noise level and the level of the correlation between the labels of the neighboring nodes, which we measure using the assortativity coefficient of Newman (2003). As our collective models, we used a pairwise Markov Random Field (MRF) (Taskar et al., 2002) and Iterative Classification Algorithm (ICA) (Lu

and Getoor, 2003a; Neville and Jensen, 2000). For `MRF` inference, we used loopy belief propagation (Yedidia et al., 2000). For `ICA`, we used the count aggregation for feature construction, and used logistic regression as the underlying classifier. We next describe the procedure we used to generate synthetic data.

4.4.1.1 Synthetic Data Generation

We generated synthetic networks using the forest-fire graph generation model (Leskovec et al., 2007b). The forest fire model is shown to exhibit many real-world phenomena such as power law degree distribution, small world effect, and shrinking diameters. However, the forest-fire method, like most random network generators, does not generate labels and attributes for the nodes. In order to label the nodes, we used the method described in (Rattigan et al., 2007b). In their method, for each label an initial number of random nodes are selected and labeled with it. Then, an each iteration, nodes that have not received a label yet are labeled based on their neighbors' labels. The number of nodes that are labeled at random at the initial phase of the algorithm controls the assortativity of the network; the higher the initial number of labelings, the less the assortativity. We varied the initial number of labelings to obtain different levels of assortativity.

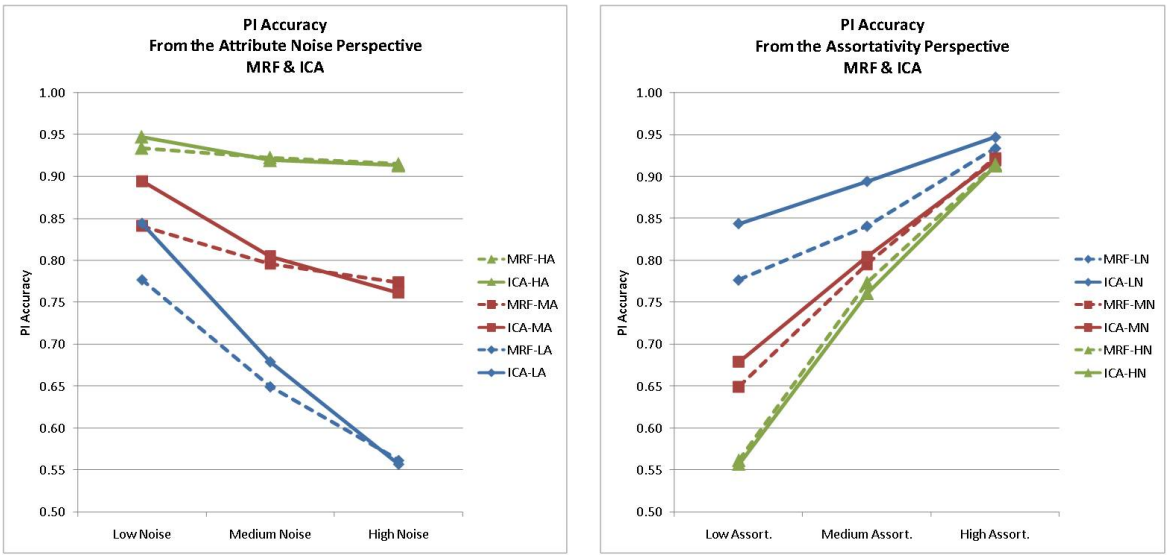
Rattigan et al. (2007b) did not generate attributes for the nodes. We generated attributes using a simple Naive Bayes model after we labeled the nodes. Assuming we have m labels, we generated $m \times k$ binary attributes, a_{ij}, where k is a parameter controlling the total number of attributes generated, i ranges from 1 to m and j

ranges from 1 to k; that is, the attributes are grouped by the value of the class label. $P(a_{ij} = True \mid class = l)$ is set to $p > 0.5$ if $i = l$, and it is set to $q < 0.5$ if $i \neq l$. The values of the attributes were sampled by conditioning on the label of the node. In our experiments, we set $m = 5$ and $k = 4$, giving us 20 attributes per node. We varied the p and q parameters to obtain different levels of attribute noise.

4.4.1.2 The Spectrum of Flooding

We generated 10 train-test graph pairs, each with 2000 nodes and with varying attribute noise and assortativity levels. We experimented with three noise levels: low, medium, and high. We distributed these levels uniformly between 0.2 (pure random classification accuracy) and 1; thus, accuracies corresponding to high, medium, and low noise levels are 0.4, 0.6, and 0.8 respectively. Similarly, we distributed the assortativity levels between 0 and 1 uniformly, having 0.25, 0.5, and 0.75 assortativity coefficients for the networks of low, medium, and high assortativity.

Before presenting the flooding percentages of `MRF` and `ICA`, we present the `PI` accuracies they achieved under varying attribute noise and assortativity level settings. The corresponding plots are shown in Figure 4.3. Not surprisingly, both `MRF` and `ICA` achieve the highest `PI` accuracies under low attribute noise or high assortativity settings. However, when the assortativity level is not high and if the attributes are not very noisy, `ICA` outperforms `MRF` in terms of `PI` accuracy. This result indicates that `ICA` is possibly better at exploiting the attribute information compared to `MRF` under comparably low assortative settings. The fact that they

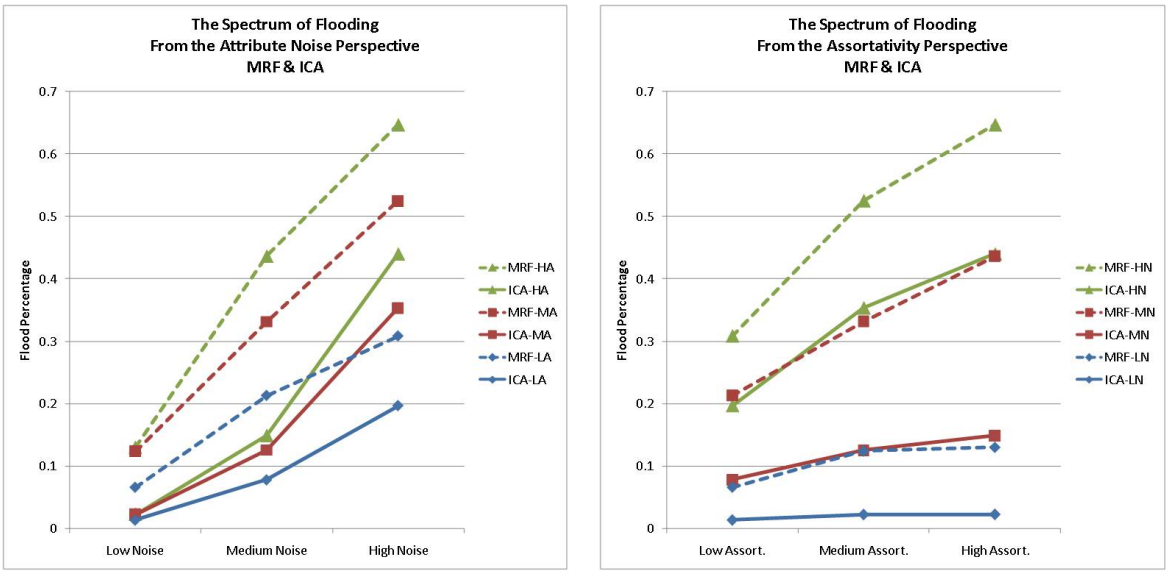


(a) (b)

Figure 4.3: Analysis of the PI accuracies achieved by MRF and ICA under low noise (LN), medium noise (MN), high noise (HN), and low assortativity (LA), medium assortativity (MA), and high assortativity (HA) settings. (a) PI accuracy from the attribute noise perspective. (b) PI accuracy from the assortativity perspective.

achieve comparable PI accuracies when the assortativity level is high regardless of the attribute noise level signals that MRF is able to exploit the neighborhood information at least as well as ICA.

We present the flood percentage results in Figure 4.4. As one expects, as the attribute noise level increases, the flood percentage also increases, and this is true for both MRF and ICA (Figure 4.4(a)). Because the attributes are noisy 1) a misclassification is more likely, 2) classification decisions depend more on the labels of the neighboring nodes, and 3) there are not many nodes that can prevent misclassification propagation (nodes whose attributes carry enough signal in the degree that they can be classified using only their attribute information). If we look at the problem from the perspective of assortativity (Figure 4.4(b)), when the attribute noise is high, the higher the assortativity, the higher the flooding; however,



(a) (b)

Figure 4.4: Analysis of the error caused by flooding, measured as the difference between PI accuracy and NOACQ accuracy, under low noise (LN), medium noise (MN), high noise (HN), and low assortativity (LA), medium assortativity (MA), and high assortativity (HA) settings. (a) Flood percentage from the attribute noise perspective. (b) Flood percentage from the assortativity perspective.

when the attribute noise is low, the assortativity does not play a significant role for flooding. Again, higher assortativity means greater dependence on the neighbors, but when the attribute noise is low, attributes also play a significant role in the classification decision. Thus, nodes are less likely to be misclassified and there are enough nodes that can prevent misclassification propagation. Finally, we observe that MRF floods more than ICA does. This observation is also in line with the observation we made earlier that ICA is able to exploit attribute information more than MRF.

With these experiments, we provide only a flavor of the different settings under which flooding occurs. Having analyzed the flooding, we next show how active inference can be used to detect and correct it.

4.4.2 Experiments Comparing Different Active Inference Techniques

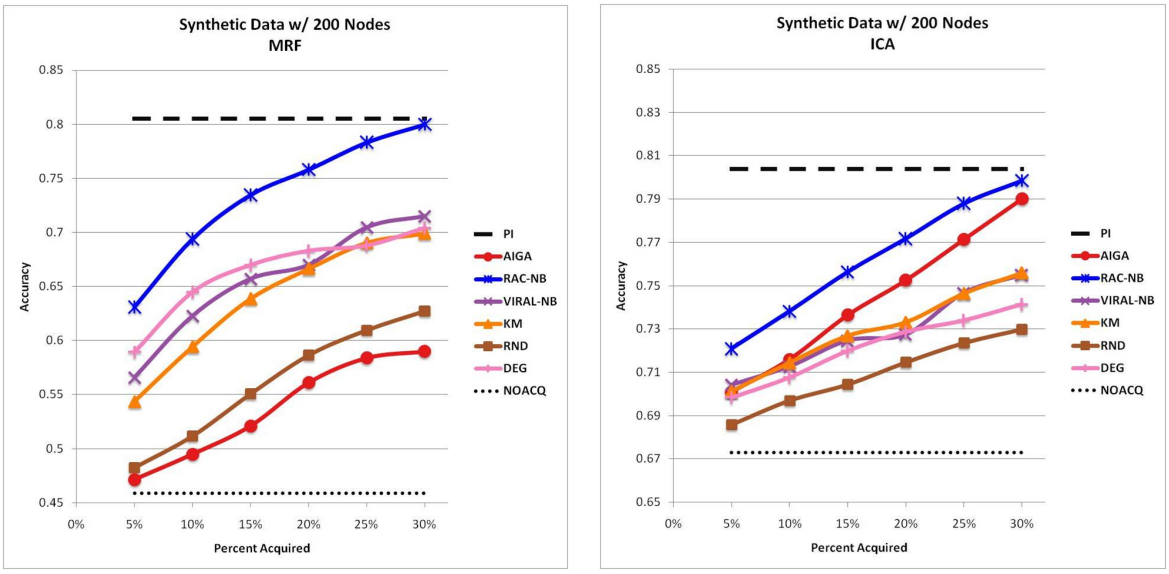
Next, we move on to the main evaluation of the proposed algorithms. We compared six acquisition methods: the three acquisition methods described earlier (`AIGA`, `VMA`, and `RAC`), two acquisition methods that are based on the structural properties of the network, and the random acquisition (`RND`) as a baseline. The two structural acquisition methods are: degree (`DEG`) and k-mediods clustering (`KM`). `DEG` ranks the nodes according to their degree in the network and acquires the highest degree ones. The intuition is that the high degree nodes affect more nodes in the network. The `KM` method, which was proposed by Rattigan et al. (2007b) and shown to outperform many other structure-based acquisition strategies, clusters the network into k clusters using k-mediods clustering and acquires the mediods of the clusters. To compute the similarity of nodes, it uses geodesic distances of nodes in the network. The intuition is to spread the acquired labels in the network evenly and to choose central nodes whenever possible.

We compare these acquisition strategies on both synthetic and real-world datasets using accuracy as the performance measure. To assess how well each acquisition methods deals with flooding, we also plot `PI` and `NOACQ` accuracies. For different acquisition methods, we computed accuracy over the labels that were not acquired (i.e. $\mathbf{Y} \setminus \mathbf{A}$), whereas for `PI` and `NOACQ`, we measured accuracy on all the labels $\mathbf{Y}$. Even though `PI` and `NOACQ` accuracies are quite useful to know to asses how well the acquisition strategies perform, neither `PI` is an upper bound nor `NOACQ` is a lower bound because the acquisition strategies and the `PI` and `NOACQ` are eval-

uated on different sets. An acquisition strategy, in practice, can acquire the labels that even PI misclassifies, thus can surpass PI accuracy. Similarly, an acquisition strategy might acquire labels that even NOACQ does not make mistakes on, thus even though both the acquisition strategy and NOACQ make the same number of mistakes, the acquisition strategy has worse accuracy than NOACQ simply because the acquisition strategy is evaluated on a smaller set ($\mathbf{Y} \setminus \mathbf{A}$ versus $\mathbf{Y}$).

Finally, both RAC and VMA require content-only classifiers, classifiers that use only the attribute and label information for each node independently; RAC requires a content-only classifier for feature construction (the content and relational indicators) and it requires a content-only classifier for predicting whether a node is misclassified given the constructed features, whereas VMA requires a content-only classifier for local probability computation. For RAC feature construction and VMA local probability computation, we used Naive Bayes for the synthetic datasets (because we generated the attributes using Naive Bayes) and we compared using Naive Bayes versus logistic regression for the real-world datasets. To classify nodes as misclassified or not based on the constructed features, we used logistic regression for both synthetic and real-world datasets.

Even though the AIGA method is a polynomial-time algorithm, each single acquisition decision requires running inference for each node and for each possible value of its label. Thus, it is impractical to run AIGA on large networks.We first present a series of results on networks small enough to run AIGA, and then present results on larger networks, which do not include AIGA results.



(a) (b)

Figure 4.5: Experiments comparing `AIGA` with other methods on small size graphs. (a) `MRF` results. (b) `ICA` results.

4.4.2.1 Experiments on Synthetic Networks with 200 Nodes

We trained and tested our collective models on ten training-testing network pairs; the training networks had 2000 nodes in order to learn a reliable collective model and the testing networks had 200 nodes. The nodes had medium noise and assortativity levels. We varied the percentage of labels acquired from 5% to 30% in 5% increments. The results are shown in Figure 4.5.

One of the striking results is that for `MRF`, `AIGA` performed worse than random (Figure 4.5(a)). This is surprising at first, because one would expect the `AIGA` method to perform the best. However, recall that we used loopy belief propagation for `MRF` inference, which is an approximate method for probability computation and it is known to produce suboptimal results when there are short cycles in the graph. Because of the assortativity of the nodes, we observed that beliefs about the

nodes' labels reinforced one another iteratively, and thus most of the probability distributions for the nodes' labels were extreme: 1 for one label value and 0 for the other values. Because the probabilities were extreme, `AIGA` made acquisition decisions based on $utility_{aiga}$ that were extremely similar for many nodes. As for `ICA` (Figure 4.5(b)), `AIGA` performed better than all methods except `RAC`. This suggests that the probabilities for `ICA` were better calibrated than for `MRF`, however, this claim needs further investigation.

As for the time it took for different acquisition methods to complete, at 30% acquisition level for `MRF`, `AIGA` took about 38 minutes, `RAC` took 4 seconds, the other methods took less than a second. For `ICA`, `AIGA` took 11 minutes, `RAC` took 3 seconds, and the other methods again took less than a second. Because `AIGA` takes much more time and its accuracy depends heavily on the calibration of the probability estimates, `AIGA` is an undesirable acquisition method. Some approaches to speeding up `AIGA` are to work with a sample rather than using all of the test data and to acquire more labels at a time, however, these modifications will most likely further reduce its accuracy.

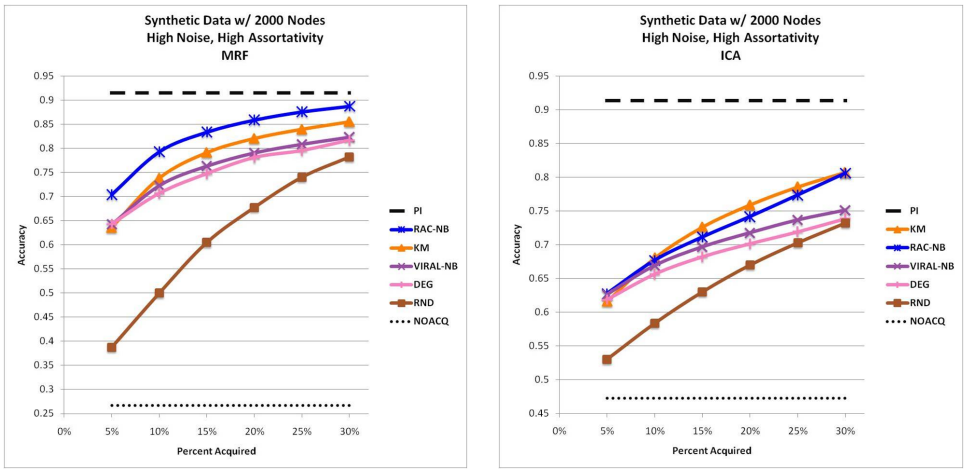
4.4.2.2 Experiments on Synthetic Networks with 2000 Nodes

Next, we compared the acquisition strategies on larger test graphs with 2000 nodes under nine settings: the cross product of the variations in the attribute noise level and assortativity level. For each plot, we zoom in to the area of interest; that is, the low point of the y-axis starts from `NOACQ` accuracy, and the highest point in the

y-axis is the accuracy of either PI or the accuracy achieved by the best performing acquisition method, whichever is higher. The reason we zoom in is to be able to highlight the differences between different acquisition methods.

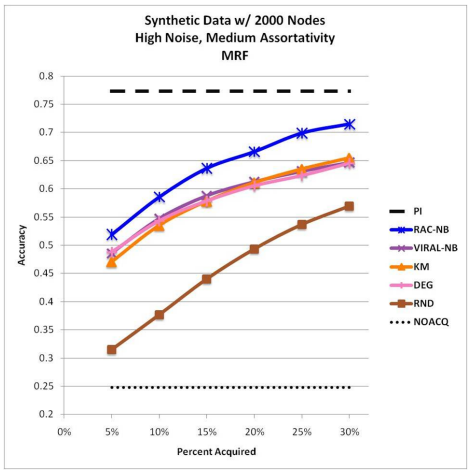
We present results in the order of high noise, medium noise, and low noise settings. For each noise setting, we vary the assortativity levels. The results for high attribute noise and varying assortativity levels for both MRF and ICA are shown in Figure 4.6. For MRF, RAC outperforms all other methods at all levels. The differences are statistically significant at all levels for high and medium assortativity level, and after 15% acquisition level for the low assortativity case, where significance is measured using paired t-test at 95% confidence level. For ICA, RAC does slightly worse than KM for the high assortativity case, comparable for the medium assortativity case, and slightly better for the low assortativity case; the differences are not statistically significant except at 20% and 30% acquisition levels for the low assortativity case. Note that two of the three features that RAC uses are based on the content-only model's predictions; even though the node attributes are quite noisy and thus the content-only model is quite unreliable, RAC is either comparable or better than the other methods in the high attribute noise case.

When we compare the remaining methods, they all significantly outperform random acquisition in almost all cases (except random has comparable results to DEG and VMA only for ICA when the assortativity is high and if we acquire 30% of the labels). For high assortativity levels, KM significantly outperforms other methods for both MRF and ICA at almost all acquisition levels and this result is in line with the findings of Rattigan et al. (2007b). For medium assortative levels, there is not a

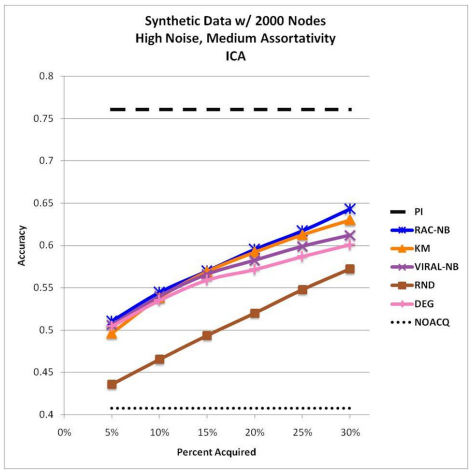


(a)

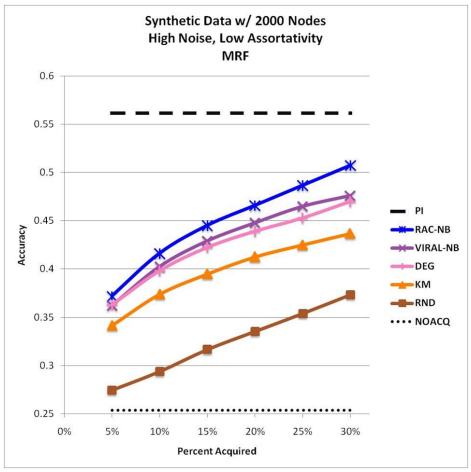
(b)



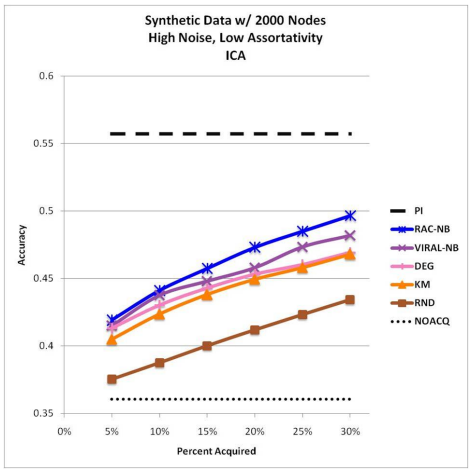
(c)



(d)



(e)



(f)

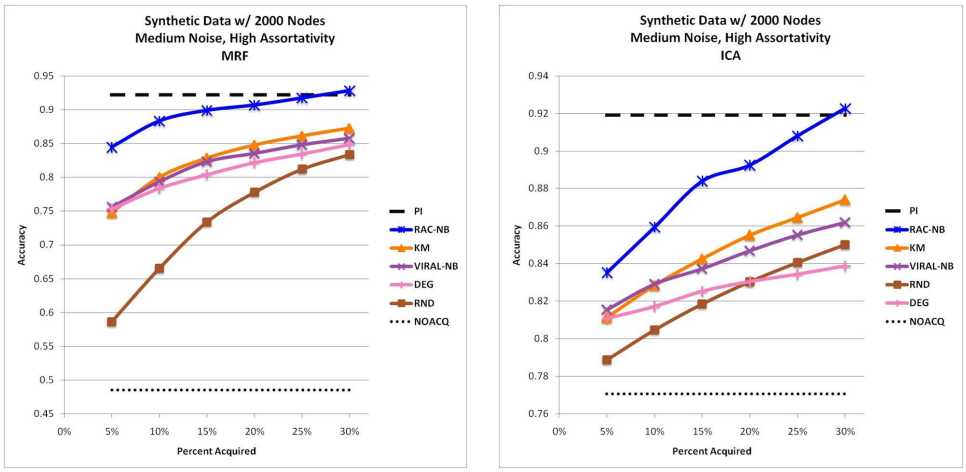
Figure 4.6: Accuracy comparisons for the high attribute noise case. (a) and (b) High assortativity, (c) and (d) Medium assortativity, (e) and (f) Low assortativity.

clear winner between VMA, KM, and DEG, but when the assortativity level is low, VMA outperforms other methods slightly.

One important observation is that MRF is easier to improve than ICA; even though MRF accuracies start very low compared to ICA due to high flood percentages, MRF accuracies are better than ICA accuracies for all acquisition methods (except RND) starting from 10% acquisition level for high assortativity and starting 15% for medium assortativity. This observation also confirms that MRF is more dependent on neighbor information than ICA.

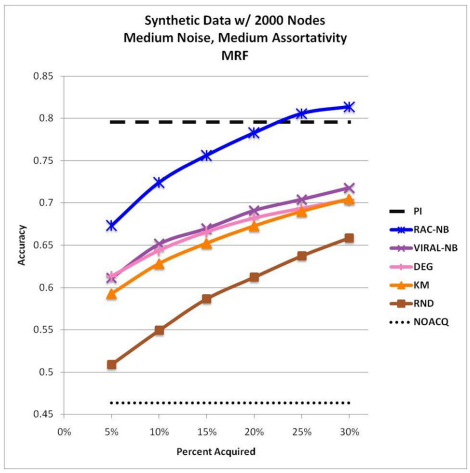
Next, we present results on the medium attribute noise case in Figure 4.7. RAC significantly outperforms all other methods at all levels for both MRF and ICA. It is also able to reach beyond PI accuracy at high acquisition levels for medium assortativity and starting 20% acquisition for low assortativity case. As for the other methods, KM again has better accuracy than other methods in the high assortativity case; for the medium assortativity and low assortativity, VMA outperforms other methods, and the differences are statistically significant for most acquisition levels.

The final set of results on low attribute noise synthetic data are shown in Figure 4.8. Remember that in this setting, the flood percentage was low, especially for ICA. Thus, the remaining errors to correct are the errors due to model imperfection, attribute noise, etc., i.e. the difference between 1 and PI accuracy, and this type of error is higher for the low assortativity case. Thus, one would expect the acquisition methods to perform better than PI accuracy in this low attribute noise setting. We observe that RAC outperforms all other methods significantly at all levels for both MRF and ICA. It is also able to perform better than PI accuracy in almost all acqui-

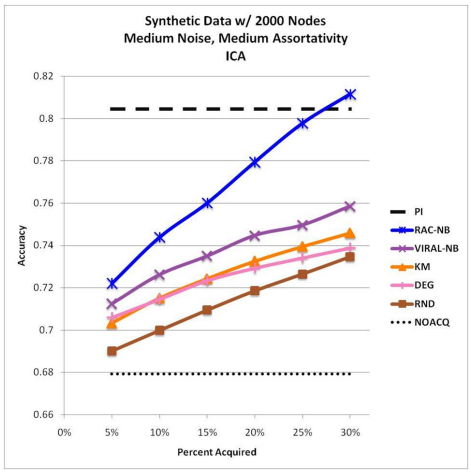


(a)

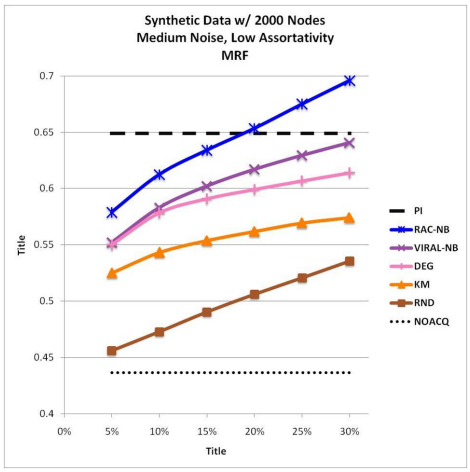
(b)



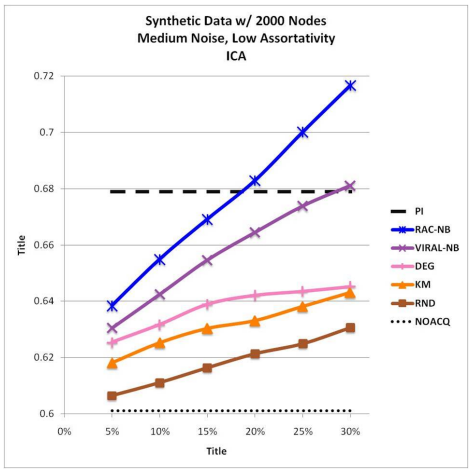
(c)



(d)



(e)



(f)

Figure 4.7: Accuracy comparisons for the medium attribute noise case. (a) and (b) High assortativity, (c) and (d) Medium assortativity, (e) and (f) Low assortativity.

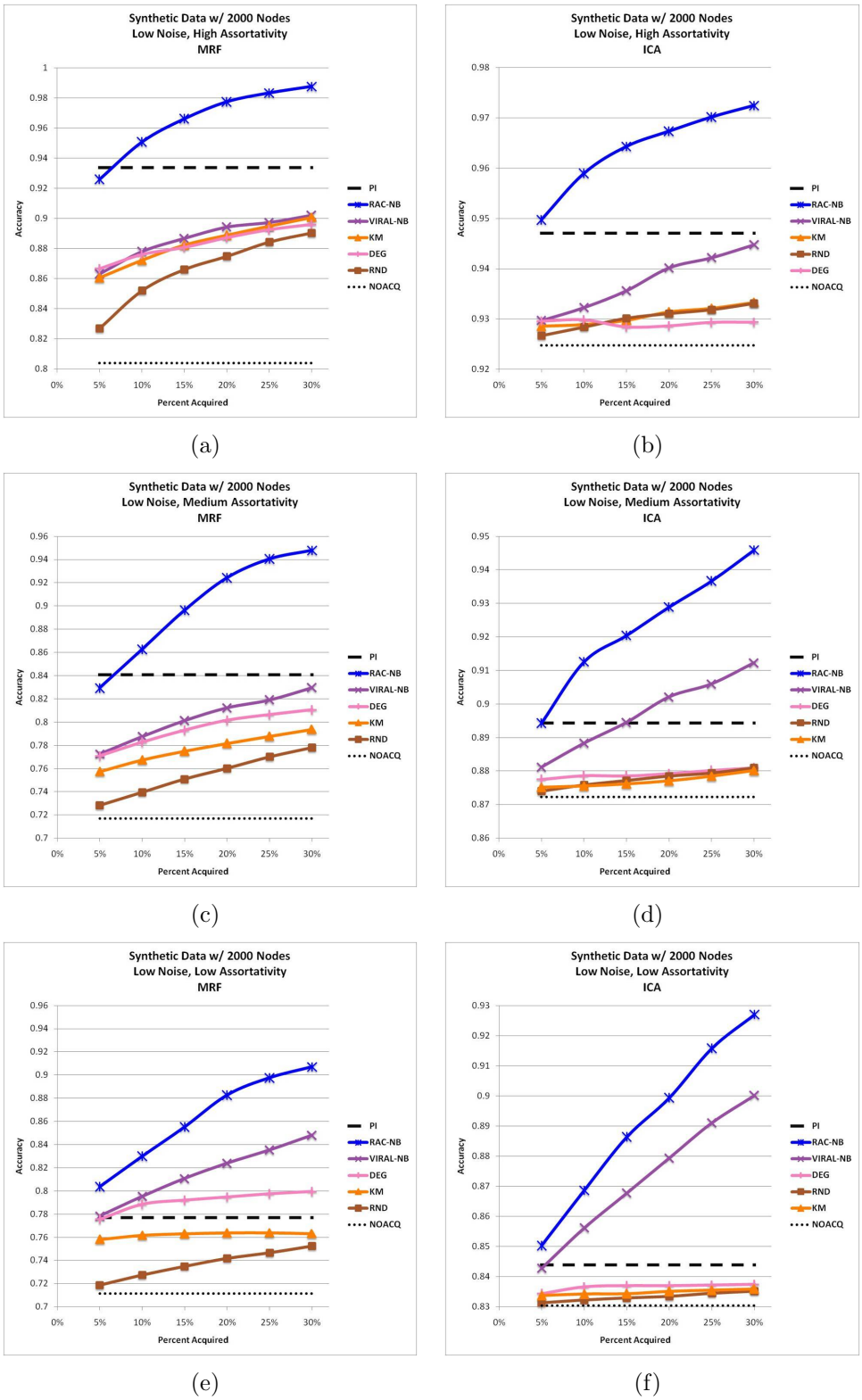


Figure 4.8: Accuracy comparisons for the low attribute noise case. (a) and (b) High assortativity, (c) and (d) Medium assortativity, (e) and (f) Low assortativity.

sition levels. As for the other methods, VMA significantly outperforms DEG, KM, and RND in most cases; it is also able to perform better than PI but only in half of the cases. This result is not surprising because VMA makes use of a content-only classifier whereas DEG, KM, and RND do not. And, for the same reason, DEG, KM, and RND are never able to get beyond PI accuracy, except DEG for only the low assortativity case for only MRF.

4.4.2.3 Experiments on Real-world Datasets

We experimented on two real publication datasets that are publicly available, the Cora dataset (McCallum et al., 2000) and the CiteSeer dataset (Giles et al., 1998). The Cora dataset contains 2708 machine learning papers that are divided into seven classes, while CiteSeer dataset has 3312 documents that are divided into six classes.

Our evaluation methodology for these datasets is slightly different from the general practice. In real-world scenarios, we typically have only a small percentage of the data labeled. This makes the interactions between the unlabeled nodes more common than the interactions between the labeled and unlabeled nodes. To mimic these two observations, we adopted the following evaluation strategy. We divided each dataset into three disjoint splits and repeatedly trained on one split and tested on the remaining two (in contrast to training on two splits and testing on the other). Additionally, we did not make use of the edges between the labeled nodes and the unlabeled ones during inference. Because of these changes in the evaluation strategy,

which we believe results in a more realistic evaluation, the accuracies corresponding to NOACQ are very low compared to the numbers reported in the literature. The primary reason is that the test graphs are more amenable to flooding now, because they are large and there are no interactions between the test graph and the training graph. However, the PI accuracies are close to the previously reported numbers (Sen et al., 2008), only being slightly lower because we are using less training data.

With these real-world datasets, we also experimented with using different content-only classifiers for RAC and VMA. The purpose of this experiment is to explore whether and how much the strength of the underlying content-only model affects the accuracy results for RAC and VMA. We experimented with using Naive Bayes and logistic regression; the content-only classification accuracies for Naive Bayes were worse than logistic regression accuracies; Naive Bayes had an average accuracy of 0.61 for Cora and 0.57 for CiteSeer, whereas logistic regression accuracies were 0.69 and 0.62 respectively. Finally, the assortativity coefficient for Cora is 0.79, whereas for CiteSeer it is slightly lower with 0.68. The accuracies for different acquisition strategies are shown in Figure 4.9.

One of the first observations is that MRF again floods more than ICA does on both datasets; the flood percentage for MRF on Cora is 0.43 and for CiteSeer is 0.35 whereas for ICA they are 0.11 and 0.06 respectively. For MRF, both versions of RAC outperform all other methods significantly for both datasets. There are not significant differences between RAC-LR and RAC-NB for MRF, except for Cora at 30% acquisition level. Because MRF floods more than ICA does, this result suggests that RAC might not need a very strong local classifier for detecting the floods. Differences

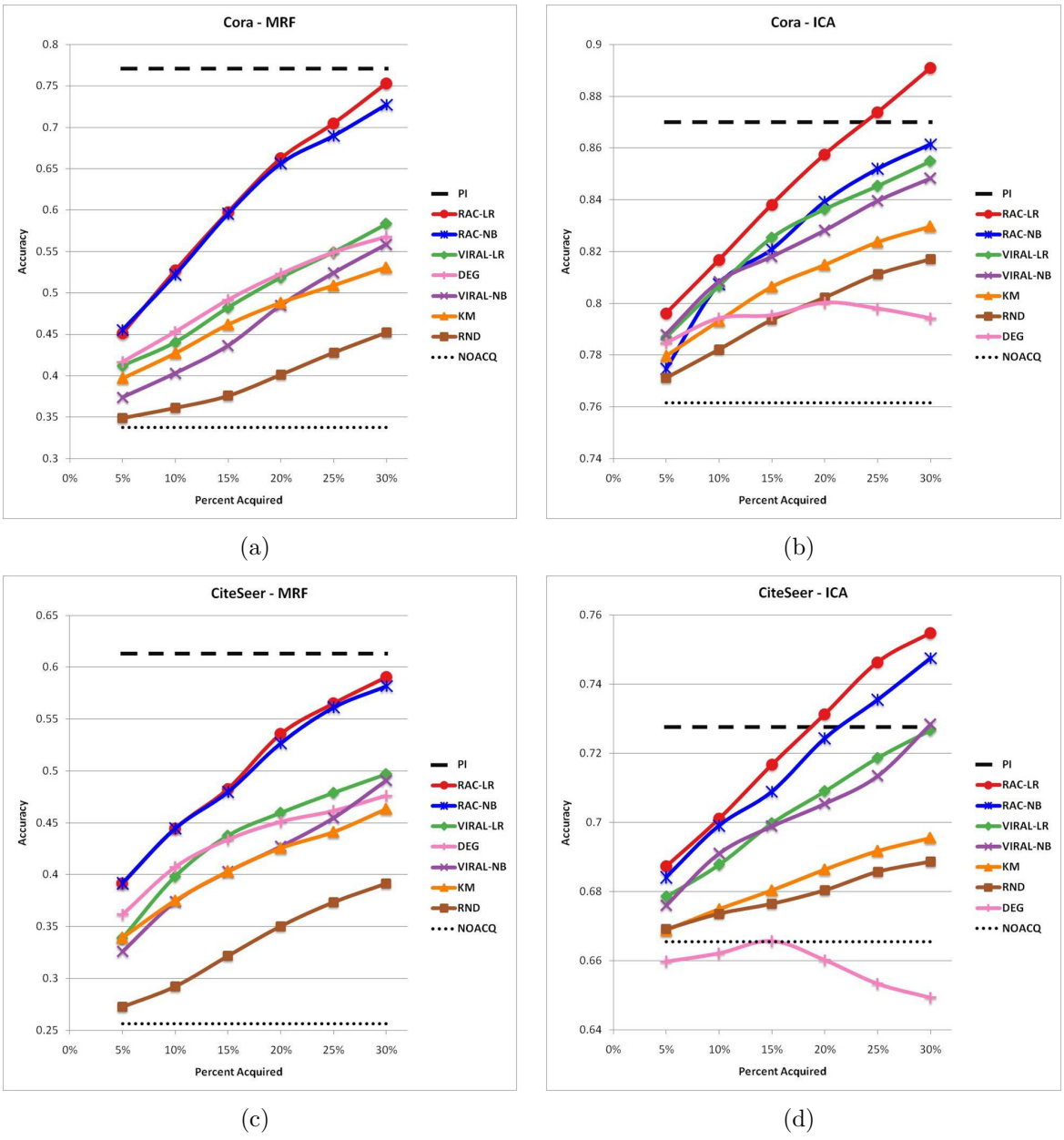


Figure 4.9: Experiments on the real-world datasets. (a) `MRF` results on Cora, (b) `ICA` results on Cora, (c) `MRF` results on CiteSeer, and (d) `ICA` results on CiteSeer.

emerge when most of the flooded nodes are corrected; `RAC-LR` outperforms `RAC-NB` significantly for `MRF` only at 30% acquisition level.

For `ICA`, `RAC-LR` outperforms all other methods, including `RAC-NB`, for both datasets. Again, this suggests that when the flood percentage is not high, which is

the case for ICA, a stronger local classifier is more helpful. RAC-NB outperforms all other methods, including VMA-LR, for CiteSeer but it has comparable performance to VMA-LR for Cora, while still outperforming the remaining methods.

One of the interesting results is that DEG performed quite poorly for ICA for both Cora and CiteSeer; for Cora, it was initially better than RND but it became worse after 20% acquisition; for CiteSeer, it was always worse than even NOACQ. This observation suggests that, at least in these real datasets, higher degree nodes can be easier to classify for ICA, and thus, acquiring labels for them is not only useless but in fact, we are investing our budget in nodes that we have a high chance of being correctly classified, which we should definitely avoid. Because we evaluated the performance of different acquisition strategies, including DEG, on the labels that were not acquired ($\mathbf{Y} \setminus \mathbf{A}$), acquiring the labels that were already classified correctly made the performance of DEG worse than NOACQ. The reason is that even though the number of misclassified nodes did not change for both DEG and NOACQ, the *percentage* of misclassified nodes increased for DEG simply because the denominator got smaller ($|\mathbf{Y}|$ for NOACQ versus $|\mathbf{Y} \setminus \mathbf{A}|$ for DEG).

4.5 Summary and Contributions

We have formulated the active inference problem in terms of expected misclassification costs and label acquisition costs and discussed why finding the optimal solution was hard under relatively general assumptions. We discussed the problem of flooding and experimentally showed that it was an important problem for two

representative collective classification models: pairwise Markov Random Field (MRF) with loopy belief propagation and Iterative Classification Algorithm (ICA). Through synthetic data, we explored the degree of flooding under varying attribute noise and label assortativity settings. We empirically showed that MRF flooded more than ICA.

We introduced three informed active inference strategies and compared them with two acquisition strategies that are based on structural properties of the network. The first informed active inference strategy that we proposed is based on approximating the value of the objective function through approximate inference and acquiring labels greedily. We experimentally showed that this method did not perform well in practice, especially for MRF. When we analyzed the reasons further, we observed that the probability estimates were not calibrated, which caused the failure of this acquisition method.

Next, we described the analogy between viral marketing and the active inference problem, and introduced a second informed active inference algorithm based on viral marketing. We showed the details of the mapping between active inference and the viral marketing formulation of Richardson and Domingos (2002). We empirically showed that this method performed equally well with the structural methods under high noise settings, and performed better as the attributes got more useful.

Of the methods that were based on structural properties of the network, the method that acquired labels according to the degree of the nodes had the most erratic performance; sometimes, it performed better than the other structure-based method, K-Mediods, while other times, it performed worse than random. K-Mediods on the other hand performed better than most acquisition methods under high noise

and high assortative settings, and it had much more stable performance compared to the degree method in the remaining settings.

Finally, we proposed a third active inference strategy called reflect and correct (RAC) that is based on learning when a collective model makes mistakes and suggests acquisitions to correct those mistakes. RAC learned a misclassification predictor using three features that we constructed by 1) comparing the predictions of a non-collective classifier and the collective model, 2) using the neighborhood information, and 3) comparing the class prior and posterior distribution statistics on the training and testing networks. We empirically showed that RAC outperformed other methods on most cases, most of the time with statistically significant differences, and it had comparable performance on the remaining few cases, never losing significantly. We also experimented with using Naive Bayes and logistic regression for constructing the features for the misclassification predictor. We showed that when the flooding was significant, RAC did not require a strong content-only classifier; otherwise, logistic regression classifier lead to better results.

4.6 Conclusions

In many real-world applications, a collective inference framework is required to make predictions. These models are often used to guide a human expert that makes the final decisions. Our work on active label acquisition helps to focus the efforts of the expert on feedback that will have the highest impact. It also highlights the complex processes involved in collective classification, and hopefully raises awareness

about the sensitivity of these models to errors, and provides some insight in how one might detect these types of errors.

Chapter 5

Label Acquisition During Learning for Relational Data

In this chapter, I introduce our approach to label acquisition during learning, i.e., active learning, for relational data. This is a very similar set up to the active inference setting I described in Chapter 4. The key difference is that in the active inference setting, we assumed that we had an already learned model and our task was to determine the right set of labels to acquire at inference time to maximize the classification performance on the remaining ones. Here, however, we assume that we are given an unlabeled network and a budget determining the number of labels to acquire. Our objective is then to determine the right set of labels to acquire and learn a classifier that is expected to have the least future misclassification cost. I describe our algorithm that uses the links in the network both to select informative instances to label and to learn a collective model.

5.1 Introduction

In many domains of interest, the instances are connected via a set of links, thus forming a network, in which neighboring instances frequently have correlated labels. For example, in document classification, documents that cite each other often have similar topics, and in social networks, people that are friends often have similar characteristics. A long tradition in machine learning has focused on exploit-

ing such network information to achieve better predictive accuracy by classifying instances *collectively*, rather than treating them as independent samples (see Sen et al. (2008) for an overview). This approach is appealing because in many cases link information is readily available. For example, in document classification, citation or hyperlinks can be automatically collected. On the other hand, labeling instances requires human attention and may be expensive. For instance, if the task is to predict the effect of a new substance on organisms in a biological network, labeling new examples may require laboratory experiments, whereas the network information regarding interactions among the organisms may be well-known.

Therefore, an important research question is to develop algorithms that reduce the amount of labeling effort required for such tasks. One promising approach is to use *active learning*. In this setting, rather than being presented with a labeled training set from the start, the learner is allowed to request labels for particular instances with the goal of achieving high accuracy with minimum number of acquired labels. While many effective active learning algorithms have been developed (see Section 2.2 for related work on active learning), to the best of our knowledge, efficient active learners that take direct advantage of explicit network structure in the data to select informative examples have not been considered.

The main contribution of this chapter is a novel active learning algorithm that addresses this setting. Our algorithm, called `ALFNET` (for active learning for networked data), exploits the network structure of the domain and the interaction between the local and relational aspects of a classifier to select more informative examples to be labeled, thus improving the accuracy of learning from fewer labeled

instances. We demonstrate the effectiveness of `ALFNET` in several real-world collective classification tasks.

Another important consideration for active learning of collective models from relational data is that learning the strength and type of correlations between the instance labels requires the labels of the linked nodes to be known during training. For example, learning the sign and strength of the correlation between the topics of linked papers in a citation network requires training data where the labels of both the citing and cited papers are known. However, because labels are scarce, it is rarely the case that labels of neighboring nodes are known. We introduce a novel semi-supervised technique that can effectively handle the problem of missing labels during training, thus providing the collective classification algorithms sufficient supervision for learning label correlations.

Further, we argue in favor of combining dimensionality reduction techniques with active learning. Even though it is well known in the literature that high dimensionality is an important problem especially when labeled data is limited, dimensionality reduction is often overlooked in the active learning community. In this chapter, we employ unsupervised dimensionality reduction as a first step of learning and show that it leads to significant performance gains.

Such semi- and unsupervised algorithms are of great importance to active learning settings in which labeled data is typically severely limited. By using them, we ensure that our proposed active learning algorithm improves over strong base learners and obtains improvements beyond those achievable by simpler methods.

The remainder of the chapter is organized as follows. In Section 5.2 we introduce some background and notation. The `ALFNET` algorithm is described in Section 5.3 and semi-supervision and dimensionality reduction is discussed in Section 5.4. We present an empricial evaluation in Section 5.5 and then conclude in Section 5.6.

5.2 Background

This section introduces necessary background and notation on collective classification and active learning. We assume that our data is represented as a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. Each node $V_i \in \mathbf{V}$ is described by an attribute vector $\vec{X}_i$ and a class label Y_i, $V_i = \langle \vec{X}_i, Y_i \rangle$. $\vec{X}_i$ is a vector of individual attributes $\langle X_{i1}, X_{i2}, \ldots, X_{ip} \rangle$. The domain of X_{ij} can be either discrete or continuous whereas the domain of the class label Y_i is discrete and denoted as $\{y_1, y_2, \ldots, y_m\}$. Each edge $E_{ij} \in \mathbf{E}$, where $E_{ij} = \langle V_i, V_j \rangle$, describes some relationship or link between V_i and V_j. For example, in a citation network, the nodes are publications, the node attributes include words, the node labels may be the topics of the papers, and the edges represent citations.

5.2.1 Collective Classification

In network data, the labels of neighboring nodes are often correlated (though not necessarily positively correlated). For example, papers that cite each other are likely to have similar topics, and proteins that interact are likely to have complementary functions. Exploiting these correlations can significantly improve classification

performance over using only the attributes, $\vec{X}_i$, for the nodes. The label correlations are typically used by defining feature functions over the labels of linked nodes and the generated features are used in addition to the local attributes to predict Y_i. However, when predicting the label of a node, the labels of the linked instances are also unknown and need to be predicted. *Collective classification* is the term used for simultaneously predicting the labels $\mathbf{Y}$ of $\mathbf{V}$ in the graph $\mathbf{G}$, where $\mathbf{Y}$ denotes the set of labels of all of the nodes, $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_n\}$.

In general, the label Y_i of a node can be influenced by its own attributes $\vec{X}_i$ as well as the labels Y_j and attributes $\vec{X}_j$ of other nodes in the graph. The variety of collective classification models that have been proposed make different modeling assumptions about how to represent and use these dependencies. Some define global objective functions, for e.g., Relational Markov Networks (Taskar et al., 2002) and Graph Mincuts (Blum and Chawla, 2001), while others such as Iterative Classification (Neville and Jensen, 2000; Lu and Getoor, 2003a), operate more locally. Here, we focus on the latter, *local collective classification models*, which consist of a collection of local vector-based classifiers, such as logistic regression, applied iteratively. For this category of collective models, each object is described as a vector of its local attributes $\vec{X}_i$ and an aggregation of attributes and labels of its neighbors. In particular, we use an Iterative Classification Algorithm (`ICA`) (Neville and Jensen, 2000; Lu and Getoor, 2003a), which we briefly explain next. However, our active learning algorithm is largely independent of the underlying collective classification model.

Let $\mathbf{N}_i$ denote the labels of the neighboring nodes of V_i, $\mathbf{N}_i = \{Y_j | \langle V_i, V_j \rangle \in \mathbf{E}\}$. A typical modeling assumption that we also make here is that, once we know the values of $\mathbf{N}_i$, then Y_i is independent of the attribute vectors $\vec{X}_j$ of all neighbors and non-neighbors, as well as of the labels Y_j of all non-neighbors.

In `ICA`, each node in the graph is represented as a vector that is a combination of node features, $\vec{X}_i$, and features that are constructed using the labels of the node's immediate neighbors, $\mathbf{N}_i$. Because nodes can have different numbers of neighbors, the size of $\mathbf{N}_i$ can vary for different i. In order to get a fixed-length vector representation, we use an aggregation function, `aggr`, over $\mathbf{N}_i$. For example, `count` aggregation constructs a fixed-size feature vector by counting the number of neighbors with each label; other examples of aggregations include `proportion`, `mode`, etc. Once the features are constructed, then an off-the-shelf probabilistic classifier can be used to learn $P(Y_i \mid \vec{X}_i, \texttt{aggr}(\mathbf{N}_i))$. Here we refer to a classifier that learns $P(Y_i \mid \vec{X}_i, \texttt{aggr}(\mathbf{N}_i))$ as `CC`, for collective classifier. We refer to a classifier that uses only the local node features and learns $P(Y_i \mid \vec{X}_i)$ as `CO`, which stands for content-only classifier.

A key component of this approach is that during inference, the labels of the neighboring instances are often not known. `ICA` addresses this issue, and performs collective classification by using the predicted labels for the neighbors for computing the aggregates. `ICA` iterates over all nodes making a new prediction based on the predictions made for the unknown labels of the neighbors in the previous iteration; in the first step of the algorithm, initial labels can be inferred based solely on attribute

information, or based on attribute and any observed neighboring labels. `ICA` is explained in more detail in Section 4.2.1.1.

5.2.2 Active Learning

Active learning addresses the problem of minimizing the labeling cost by letting the underlying classifier choose which examples to label. A variety of active learning settings have been studied, such as pool-based sampling, membership query synthesis, and stream-based selective sampling (see Settles (2009) for an overview). In this chapter, we consider the pool-based setting, in which the learner is initially provided with a pool of unlabeled examples $\mathbf{P}$. At each step, it is allowed to select a batch of k instances that are added to its labeled corpus $\mathbf{L}$ and removed from $\mathbf{P}$. We utilize and build upon uncertainty sampling (Lewis and Gale, 1994), committee-based sampling (Seung et al., 1992), and clustering (Dasgupta and Hsu, 2008). A more thorough discussion of related work is provided in Section 2.2.

5.3 `ALFNET`

`ALFNET` is a novel active learning algorithm for collective classification. Before describing it in detail, we provide a precise statement of the problem we study.

Problem Statement: We are given a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where a subset $\mathbf{P} \subset \mathbf{V}$ is the pool of unlabeled examples, a collective classification model `CC` to be trained, a batch size k, and a budget B. The task is, within the constraints of B, to make a series of selections of k elements from $\mathbf{P}$ to be labeled by an oracle so

that the accuracy of CC on unseen data, after training it on the acquired labeled examples $\mathbf{L}$, is maximized.

This is an inductive set-up, in which the test data, $\mathbf{V} \setminus \mathbf{P}$, is *not* available during the active learning process, i.e., testing is done on unseen instances and not on the remaining part of the pool $\mathbf{P}$. However, we assume that the labeled data and the remaining unlabeled instances are available at test time. In the remainder of this section, for simplicity of notation we assume that the test nodes and their adjacent edges have been removed from the training graph $\mathbf{G}$, and so, the initial pool consists of all nodes in $\mathbf{V}$.

The difference between the problem addressed in this and previous active learning approaches is that here we assume that the instances to be classified form a network structure, as defined by the edge set $\mathbf{E}$ of the graph $\mathbf{G}$. ALFNET can take advantage of this additional information to select more informative instances. It proceeds by first using the network structure to cluster the data. It then requests the labels of examples that belong to clusters in which CC and CO (1) disagree about the class assignments of the yet unobserved instances and (2) make predictions that do not match the majority label among the observed ones in the cluster.

The high-level pseudo code for ALFNET is described in Algorithm 5.1. First, in line 2, the network structure, as given by the edge set $\mathbf{E}$, is used to cluster the nodes of $\mathbf{G}$ into at least k clusters, where k is the batch size, as defined above. To obtain initial data for training the base learner, k clusters are selected, and one item from each of them is picked and labeled (lines 3-6). This forms the initial labeled set $\mathbf{L}$. ALFNET then proceeds in iterations until the budget B is exhausted (lines 7-15), as

Algorithm 5.1: `ALFNET`: Active Learning for Networked Data

Input: $\mathbf{G} = (\mathbf{V}, \mathbf{E})$: the network, `CO`: content-only learner, `CC`: collective learner, k: the batch size, B: the budget
Output: $\mathbf{L}$: the training set

```
1  L ← ∅
2  C ← Cluster the nodes V of the network G into at least k clusters
3  C^k ← Pick k clusters from C
4  foreach Cluster C_i ∈ C^k
5      V_j ← Pick an item from C_i
6      Add V_j to L
7  while |L| < B
8      Re-train CO and CC
9      foreach Cluster C_i ∈ C
10         score(C_i) ← Disagreement(CC,CO,C_i, L)
11     C^k ← Pick k clusters based on the scores
12     foreach Cluster C_i ∈ C^k
13         V_j ← Pick an item from C_i ∩ P
14         Add V_j to L
15         Remove V_j from P
```

follows. Although only the accuracy of `CC` is tested in the evaluation, both `CC` and `CO` are trained in parallel so that their predictions can be compared for the purposes of computing a disagreement score. In each iteration, `CO` and `CC` are re-trained using the currently labeled data $\mathbf{L}$ (line 8). For each cluster, `ALFNET` computes a score of the disagreement of `CO`, `CC`, and the observed labels in the cluster, and selects k clusters based on their scores. We provide details on how the disagreement score is computed later in this section. One unlabeled item from each of the selected clusters is labeled, added to $\mathbf{L}$, and removed from $\mathbf{P}$ (lines 13-15).

Next, we provide more detail on how the clusters are computed, how clusters and elements from them are selected, and how the disagreement score is calculated.

For each of these, a variety of options can be explored. Here we focus on the choices made in our implementation.

Clustering the nodes (step 2): There are many options on how to cluster the nodes $\mathbf{V}$ of the graph $\mathbf{G}$. While in previous work, (Dasgupta and Hsu, 2008), clustering was performed based only on object attributes, here, we take advantage of the available network structure and use a graph clustering algorithm to find clusters. For our experiments we chose modularity clustering by Newman (2006). The algorithm was allowed to split larger clusters into sub-clusters until one of two conditions was met: splitting the cluster further either did not add to the modularity score (Newman, 2006), or it would result in clusters with size smaller than a pre-defined threshold θ. In the experiments, we set $\theta = 200$ and did not consider other values. Clustering the nodes based on network structure is promising because it identifies groups of related nodes in the data, and thus helps the active learner obtain a balanced training set, while avoiding areas of the data for which sufficient supervision is already acquired.

Computing the disagreement score of a cluster (step 10): Intuitively, the disagreement score of a cluster $\mathbf{C}_j$, captures the degree to which `CC` and `CO` differ in their predictions from each other, as well as the majority class, `MC`, of observed labels in the cluster. The overall disagreement score of $\mathbf{C}_j$ is defined as the sum of the local disagreement (`LD`) scores for each unlabeled node in cluster C_j, divided by

the number of labeled nodes in the cluster:

$$\texttt{Disagreement}(\texttt{CC}, \texttt{CO}, \texttt{MC}, \mathbf{C}_j) = \frac{1}{|\mathbf{C}_j \cap \mathbf{L}| + 1} \sum_{V_i \in \mathbf{C}_j \cap \mathbf{P}} \texttt{LD}(\texttt{CC}, \texttt{CO}, \texttt{MC}, V_i). \quad (5.1)$$

Dividing by the number of labeled instances in the cluster is needed to avoid over-investing in the clusters that have already been explored.

To define the local disagreement LD for an unlabeled node V_i, we collect the predictions of three classifiers, regarding the label of V_i. The first two are the most likely labels predicted by CC and CO, respectively, and the third one is the majority class, MC, in the already observed nodes in $\mathbf{C}_j \cap \mathbf{L}$. We form a weighted vote classifier, WVC, using these three predictions:

$$P_{\texttt{WVC}}(Y = y_j \mid \texttt{CC}, \texttt{CO}, \texttt{MC}) = \frac{w_{\texttt{CC}} \times \delta(p_{\texttt{CC}} = y_j) + w_{\texttt{CO}} \times \delta(p_{\texttt{CO}} = y_j) + w_{\texttt{MC}} \times \delta(p_{\texttt{MC}} = y_j)}{w_{\texttt{CC}} + w_{\texttt{CO}} + w_{\texttt{MC}}} \quad (5.2)$$

where $w_{\texttt{CC}}$ is weight of the CC classifier, $p_{\texttt{CC}}$ is the prediction made by CC, and δ is the dirac delta function. $w_{\texttt{CO}}$, $p_{\texttt{CO}}$, $w_{\texttt{MC}}$ and $p_{\texttt{MC}}$ are defined similarly. The local disagreement LD of a node V_i is defined as the entropy of V_i's label according to the class distribution $P_{\texttt{WVC}}$:

$$\texttt{LD}(\texttt{CC}, \texttt{CO}, \texttt{MC}, V_i) = \mathrm{H}_{P_{\texttt{WVC}}}(V_i). \quad (5.3)$$

Therefore, the more diverse the predictions of the three classifiers are, the greater the disagreement about an instance is.

Picking clusters (steps 3 and 11): ALFNET picks k of the clusters, from which it selects items for labeling. In general, the clusters may differ in size, and

thus the cluster sizes should be taken into account. In step 3 of the algorithm, k clusters are picked probabilistically in proportion to their sizes. In step 11, the top k clusters are picked, where clusters are sorted according to their disagreement scores.

Picking an item from a given cluster (step 5 and 13): Given a cluster, an item is chosen randomly at step 5, and the item with the highest local disagreement `LD` score is picked for labeling at step 13.

5.4 Semi-supervision and Dimensionality Reduction

An important aspect of active learning for networked data is that the collective classification algorithms need access to the labels of linked nodes in order to learn the label correlations in the network. More specifically, the collective classifier is trained on the combined local features $\vec{X}_i$ and features constructed using the neighborhood $\mathbf{N}_i$. For example, the collective component of `ICA` is trained on $(\vec{X}_i, \texttt{aggr}(\mathbf{N}_i))$, where $\texttt{aggr}(\mathbf{N}_i)$ is computed *only* over neighbors for which observed labels are available. When labeled data is scarce, there is an insufficient number of observed neighbors. We introduce a novel semi-supervised collective classification method, which is simple, but quite effective, as we show in the experiments. In this technique, `CO` is used to predict labels for the unobserved members of $\mathbf{N}_i$, i.e., $\mathbf{N}_i \setminus \mathbf{L}$. The aggregation function $\texttt{aggr}(\mathbf{N}_i)$ is then computed over actual (predicted) labels for the observed (unobserved) neighbors. This results in much stronger supervision for the neighborhood features.

Further, when the number of local features is large and the size of **L** is not big enough, then there is not enough supervision to learn the model parameters reliably. Because the size of **L** is bound to be small in the active learning scenario, we argue for combining dimensionality reduction techniques with active learning. Since there is not enough supervision, we employ unsupervised dimensionality reduction as a first step of learning. Specifically, we used principal component analysis (PCA) to transform the original feature space into a smaller one, over which learning from less data is more effective. We experimentally show that this leads to significant performance gains.

5.5 Experiments

We experimented with `ALFNET` in two benchmark collective classification tasks. Our experimental study is structured as follows. First, we use the techniques described in Section 5.4 to strengthen the base learner. We then compare the accuracy of `ALFNET` to that of several competitive baselines. We then perform an ablation study in which we test the importance of different aspects of `ALFNET`. Finally, we discuss various strategies for weighting the votes of `CO`, `CC`, and `MC`.

5.5.1 Data

We experimented with two real-world publication data sets – Cora and CiteSeer, prepared by Sen et al. (2008)[1]. Cora contains 2708 instances, each belonging to one of seven classes, while CiteSeer contains 3312 instances, each of which is in

[1]The datasets are available at `http://www.cs.umd.edu/projects/linqs/projects/lbc/`.

one of six classes. In both data sets, instances correspond to documents and are described as 0/1 feature vectors, which indicate the absence/presence of a word. The size of the vocabulary in Cora is 1433 and CiteSeer is 3703 words. The network structure of both domains is provided by the citations between documents. Cora contains 5429 citation links, while CiteSeer contains 4732. We ignored the direction of the links, treating two documents as connected if either of them cited the other. In a preliminary analysis, we discovered that in each of the data sets one connected component contained a large percentage of all documents, whereas the remaining documents were sparsely connected in components of average size 2.86/2.75 for Cora/CiteSeer. We attribute this sparsity to missing information in the data. Because in this work we are interested in collective classification, and thus the presence of links between the documents is essential, we cleaned up the data by removing instances that do not belong to the largest connected component. In this way, we were left with 92% of all instances in Cora and 64% in CiteSeer.

5.5.2 Methodology

We performed 10-fold cross-validation by randomly partitioning the data into 10 pieces. During training, in each fold of cross-validation, the instances from one of the partitions were held for testing, and all of their links to the rest of the data were removed to avoid contaminating the test set. The instances from the remaining nine partitions had their labels hidden and constituted the pool $\mathbf{P}$, from which the active learner selected $k = 5$ instances to be labeled and added to the training set

in each iteration. During testing, the links between the test instances and the rest of the data were restored. Data labeled during the learning stage was available during testing. However, to ensure that all systems were tested on the same set of examples, we evaluated their accuracy only on the held-out test set, and not on unlabeled examples from $\mathbf{P}$. In each fold, we performed three runs for each of the systems; thus, each point on the learning curves presented here is an average of 30 runs. For the weighted vote classifier (Equation (5.2)) for cluster $\mathbf{C}_j$, we set $w_{\texttt{CC}} = 1$, $w_{\texttt{CO}} = 1$, and $w_{\texttt{MC}} = \max(1 - \mathit{Entropy}(\mathbf{C}_j \cap \mathbf{L}))$. By discounting the vote of `MC` by the entropy of the observed label distribution in the cluster, we make sure that the homogeneous clusters are weighed more than the heterogeneous ones. Thus, `ALFNET` can benefit from clustering techniques that provide more homogeneous clusters, and it is protected from the negative effects of heterogeneous clustering. We further discuss various weighting strategies in Section 5.5.3.4.

The base classifier used for `CO` and `CC` was logistic regression (LR). During preliminary experiments with these data sets, we additionally experimented with SMO and Naive Bayes, and selected LR as the best among the three. For aggregating the label information from the neighboring nodes for `CC`, we used `proportion` where for each class, we take the proportion of neighbors of V_i belonging to that class.

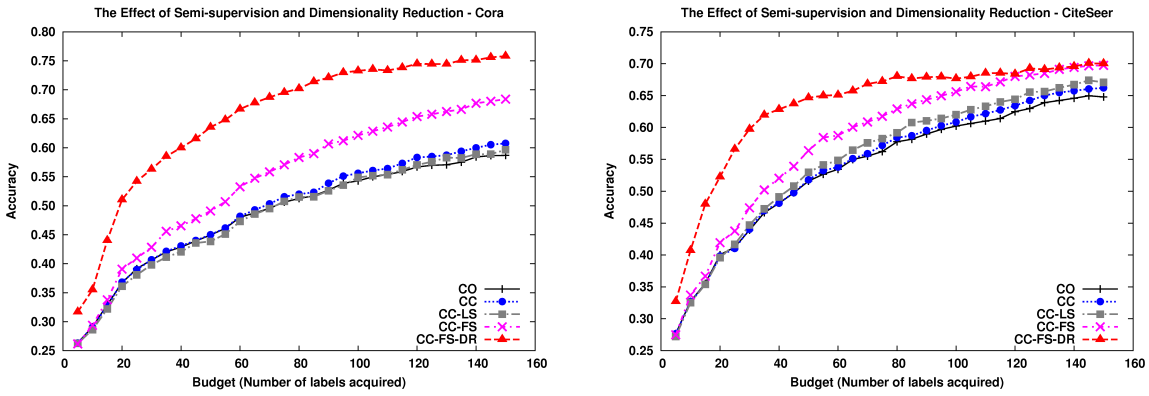
5.5.3 Results

5.5.3.1 Semi-supervision and Dimensionality Reduction

In this section, we present results on the effect of semi-supervision and dimensionality reduction. We show results for:

- `CO` – Regular content-only classification.
- `CC` – Regular collective classification (no semi-supervision).
- `CC-LS` – Collective classification with label semi-supervision, where `CO` is used to predict the labels for every instance in $\mathbf{P} \setminus \mathbf{L}$ and the collective classification model is trained using all the observed and predicted labels.
- `CC-FS` – Collective classification with feature semi-supervision, where `CO` is used to predict the labels for only the unobserved neighbors of $\mathbf{L}$ and these predicted labels are used only to generate the aggregate features for the observed ones.
- `CC-FS-DR` – Dimensionality reduction (PCA) is added on top of `CC-FS`. In our experiments, we reduced the dimensionality to 100.

We present the results in Figures 5.1(a) and 5.1(b) for Cora and CiteSeer respectively. The regular `CC` has very similar results to `CO` when the labeled data is very scarce and `CC` wins only when the budget is high. This result is due to lack of enough supervision for `CC` to learn and exploit the label correlations. What might be surprising is that `CC-LS` hurts slightly for Cora and helps slightly for CiteSeer.



(a) (b)

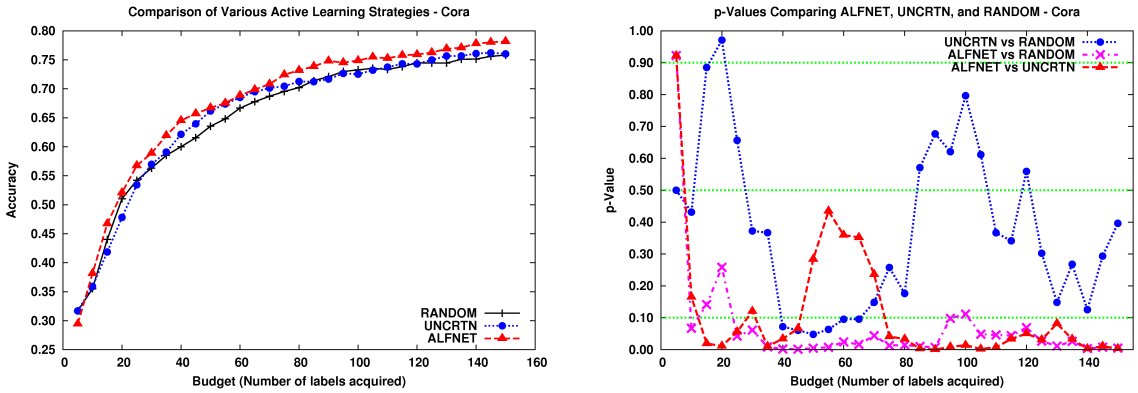
Figure 5.1: The effect of semi-supervision and dimensionality reduction. a) Cora, b) CiteSeer.

`CC-FS`, on the other hand, has clear advantages over `CO`, `CC`, and `CC-LS`. Adding dimensionality reduction on top of feature semi-supervision, `CC-FS-DR`, boosts the performance even further.

Although the issues considered in the above experiments are orthogonal to the main contribution of this work, we emphasize their importance as a means of ensuring that any improvements obtained by active learning are not over a weak "strawman," but over a carefully selected base learner that already reaches almost optimal accuracy, as reported by Sen et al. (2008), and is very challenging to improve upon. These experiments also provide strong empirical evidence in favor of coupling semi- and unsupervised techniques with active learning. For the remaining experiments, we use `CC-FS-DR` as the base learner and perform active learning using this classifier.

5.5.3.2 Active Learning

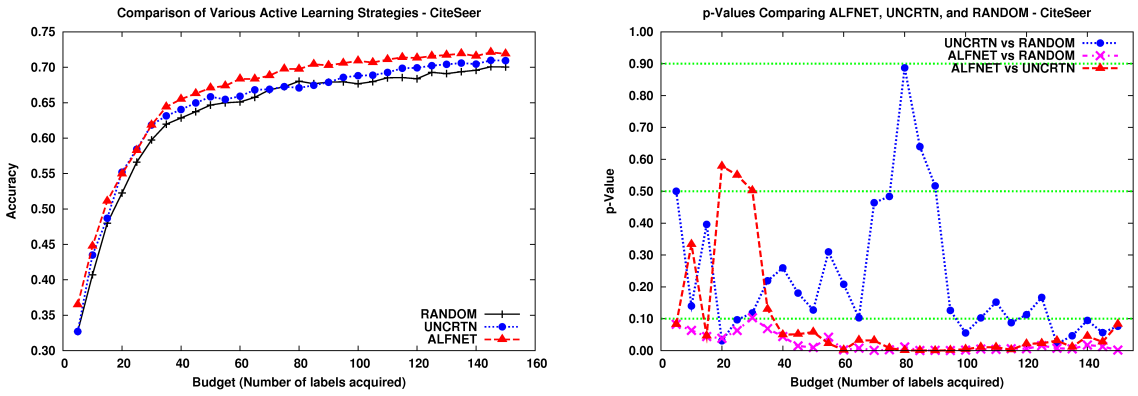
In this set of experiments, we compare the accuracy of ALFNET to that of two baselines– RANDOM, which randomly selects examples to be labeled, and Uncertainty sampling (UNCRTN), which selects the instances about whose labels CC-FS-DR is most uncertain (Lewis and Gale, 1994). UNCRTN in our experiments is measured as the expected conditional error of CC-FS-DR. To pick k items in each batch, we follow Saar-Tsechansky and Provost (2004) and use the uncertainties to weight the samples and then probabilistically choose k items. This contrasts with picking the top k most uncertain items, which is known to perform poorly (Lewis and Gale, 1994; Saar-Tsechansky and Provost, 2004).



(a) (b)

Figure 5.2: a) Relative accuracy of ALFNET in Cora. b) P-values of a paired t-test between pairs of systems in Cora. A detailed description is in the text.

We present two figures for each data set; the first one shows the accuracies of the different active learning systems, whereas the second one shows the p-values of paired t-tests between couples of systems. The accuracy results and the p-values are shown in Figures 5.2(a) and 5.2(b) for Cora and in Figures 5.3(a) and 5.3(b) for



(a) (b)

Figure 5.3: a) Relative accuracy of ALFNET in CiteSeer. b) P-values of a paired t-test between pairs of systems in CiteSeer. A detailed description is in the text.

CiteSeer respectively. Figures 5.2(b) and 5.3(b) are organized as follows. The X-axis matches the X-axis of the corresponding accuracy graph. For a curve labeled A vs. B, any point that falls below the bottom dashed green line indicates a significant win of system A, and any point above the top dashed green line indicates a significant win of system B. For example, for the curve labeled ALFNET vs. UNCRTN, ALFNET is significantly better than UNCRTN at points below the bottom dashed green line, and UNCRTN is significantly better at points above the top dashed green line. We also summarize the number of significant wins, significant loses, and ties (no significant differences) in Table 5.1.

One of the first observations is that, RANDOM is known to be very competitive for LR (Schein and Ungar, 2007) and we also observe that even though UNCRTN improves over RANDOM, the differences are not statistically significant for most budget levels for both datasets as the p-values in Figures 5.2(b) and 5.3(b) and counts in Table 5.1 show; out of 30 cases, UNCRTN wins over RANDOM only 6 times in Cora and 9 times in CiteSeer.

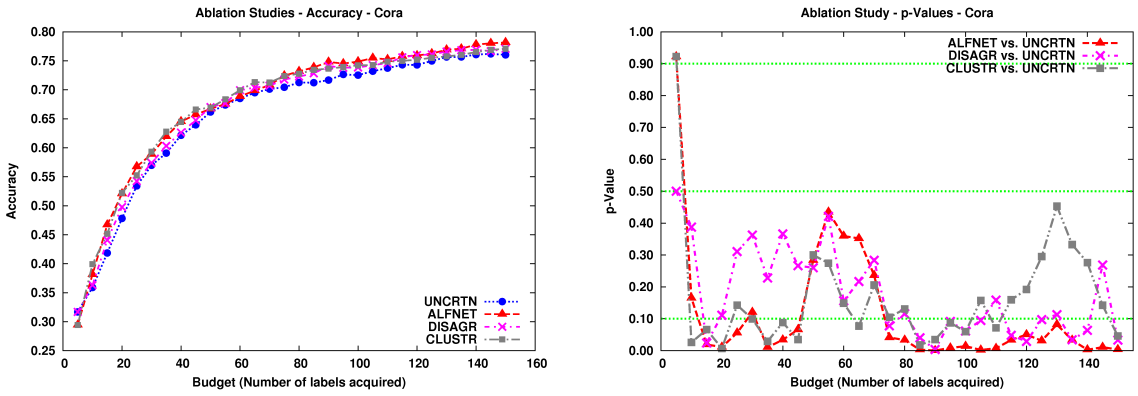
Table 5.1: p-values comparing the accuracies of ALFNET, UNCRTN, and RANDOM summarized as Win, Tie, and Loss. A $p < 0.1$ is considered a Win, $0.1 \leq p \leq 0.9$ is counted as a Tie, and $p > 0.9$ is a Loss for A when comparing A vs. B.

Dataset	Comparison	Win	Tie	Loss
Cora	UNCRTN vs. RANDOM	6	23	1
	ALFNET vs. RANDOM	26	3	1
	ALFNET vs. UNCRTN	22	7	1
CiteSeer	UNCRTN vs. RANDOM	9	21	0
	ALFNET vs. RANDOM	29	1	0
	ALFNET vs. UNCRTN	25	5	0

The ALFNET algorithm, on the other hand, improves significantly over both RANDOM and UNCRTN for both Cora and CiteSeer. ALFNET wins significantly over RANDOM 26 times in Cora and 29 times in CiteSeer. It wins significantly over UNCRTN 22 times in Cora and 25 times in CiteSeer.

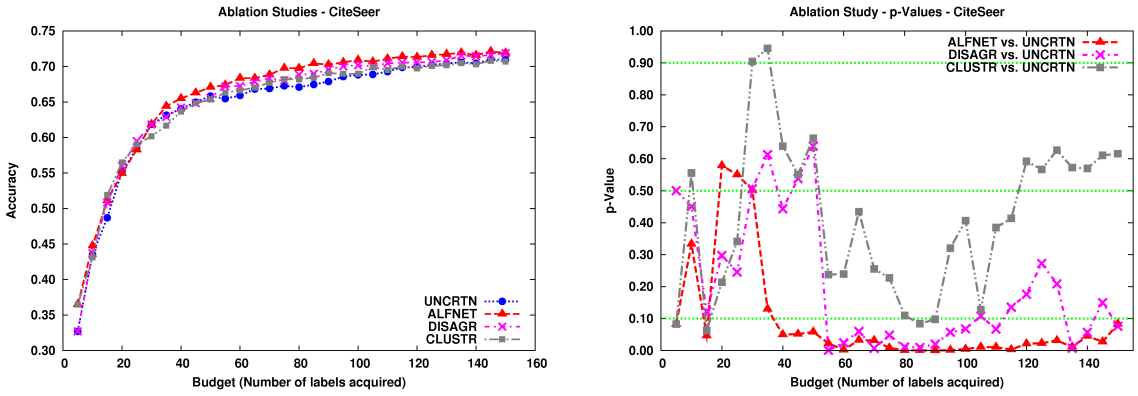
5.5.3.3 Ablation Studies

In this section, we test the contribution of each of ALFNET's subcomponents by comparing the complete ALFNET to two variants. The first one, disagreement (DISAGR), utilizes the disagreement between CO and CC, but does not exploit the cluster structure of the data. The second variant, clustering (CLUSTR), pre-clusters the data but selects the instances randomly from each cluster, rather than using disagreement. We present the accuracy and p-value graphs in Figures 5.4 and 5.5 for Cora and CiteSeer respectively. We also summarize the number of Wins, Ties, and Loses in Table 5.2.



(a) (b)

Figure 5.4: a) Ablation study - Accuracies in Cora. b) P-values of a paired t-test between pairs of systems in Cora.



(a) (b)

Figure 5.5: a) Ablation study - Accuracies in CiteSeer. b) P-values of a paired t-test between pairs of systems in CiteSeer.

These results show that `ALFNET` performs better than any of its subcomponents applied alone. For example, compared to `UNCRTN`, `DISAGR` wins 13 times and `CLUSTR` wins 14 times, while `ALFNET` wins 22 times in the Cora dataset. An important observation is that even though `CLUSTR` wins only 4 times over `UNCRTN` in the CiteSeer dataset, adding clustering on top of `DISAGR` (i.e., making it `ALFNET`), boosts the wins from 14 to 25. This result shows that even though `CLUSTR` does not perform very well alone, it helps a lot when combined with `DISAGR`.

Table 5.2: Ablation study - p-values comparing the accuracies of ALFNET, UNCRTN, DISAGR, and CLUSTR summarized as Win, Tie, and Loss.

Dataset	Comparison	Win	Tie	Loss
Cora	DISAGR vs. UNCRTN	13	17	0
	CLUSTR vs. UNCRTN	14	15	1
	ALFNET vs. UNCRTN	22	7	1
CiteSeer	DISAGR vs. UNCRTN	14	16	0
	CLUSTR vs. UNCRTN	4	24	2
	ALFNET vs. UNCRTN	25	5	0

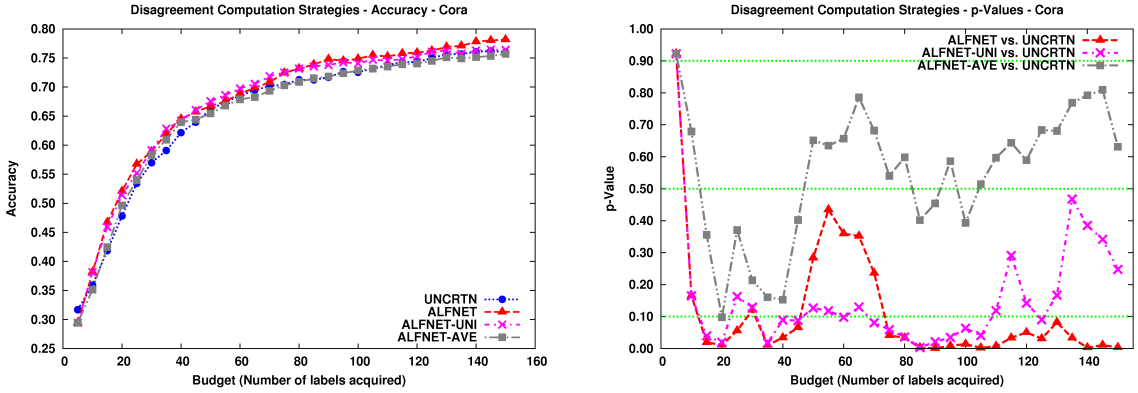
5.5.3.4 Disagreement Computation Strategies

We finally experimented with three possible ways of computing the disagreement of CO, CC, and MC. These are:

- Average probabilities (ALFNET-AVE): Take a simple average of the probabilities estimated by CO, CC, and MC, and use the entropy of this probability distribution as the disagreement score of the instance. CO and CC use logistic regression, and we use the probability estimates directly. For MC, we use the class distribution in the cluster as an estimate of the class probabilities.

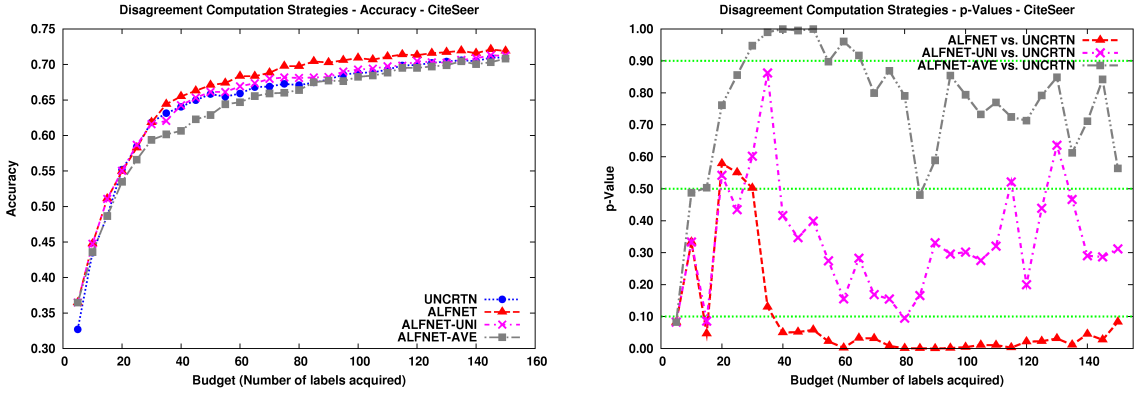
- Uniform weights (ALFNET-UNI): Rather than taking the average of the probability estimates, construct a weighted vote classifier by taking the votes of CO, CC, and MC, as described more formally in Equation (5.2). For ALFNET-UNI, we set the weights $w_{\texttt{CO}}$, $w_{\texttt{CC}}$, and $w_{\texttt{MC}}$ uniformly. The disagreement score of an instance is than the entropy of the probability distribution estimated by this weighted vote classifier (Equation (5.3)).

- `ALFNET`: Set the weights in Equation (5.2) to be as follows: $w_{\texttt{CO}} = w_{CC} = 1$ and $w_{\texttt{MC}} = \max(1 - Entropy(\mathbf{C}_j \cap \mathbf{L}))$. The motivation is that because clusters are not equally homogeneous in terms of the node labels, we discount the vote of `MC` in more heterogeneous clusters. This is the `ALFNET` method we used in the experiments that we discussed in the above sections.



(a) (b)

Figure 5.6: a) Disagreement computation strategies - accuracies in Cora. b) P-values of a paired t-test between pairs of systems in Cora.



(a) (b)

Figure 5.7: a) Disagreement computation strategies - accuracies in CiteSeer. b) P-values of a paired t-test between pairs of systems in CiteSeer.

We show the results in Figures 5.6 and 5.7 and summarize them in Table 5.3. These results show that a simple combination of probabilities through averaging

them (ALFNET-AVE) performs the worst. Moreover, applying a simple discounting factor for MC's vote provides significant performance gains over using uniform weights (ALFNET vs. ALFNET-UNI).

Table 5.3: Disagreement computation strategies - p-values comparing the accuracies of ALFNET, UNCRTN, ALFNET-UNI, and ALFNET-AVE summarized as Win, Tie, and Loss.

Dataset	Comparison	Win	Tie	Loss
Cora	ALFNET-UNI vs. UNCRTN	14	15	1
	ALFNET-AVE vs. UNCRTN	1	28	1
	ALFNET vs. UNCRTN	22	7	1
CiteSeer	ALFNET-UNI vs. UNCRTN	3	27	0
	ALFNET-AVE vs. UNCRTN	1	22	7
	ALFNET vs. UNCRTN	25	5	0

5.6 Conclusion

Active learning, semi-supervised learning, and collective classification are all important concepts within machine learning. In this work, we have shown how all of them can be leveraged in the setting where we have network data. We developed an algorithm, ALFNET, which leverages network structure in a variety of ways to select samples for labeling in an informed manner. We show how to adapt classic active learning ideas such as disagreement and clustering to the setting in which we have network structure as well as attribute information. In addition, we show how to significantly boost the baseline performance of our active learner by combining dimensionality reduction with a carefully designed semi-supervised learning. We have performed an extensive experimental evaluation and showed that principled use of

structure using ALFNET provided significant improvements over RANDOM and UNCRTN. Through ablation studies, we have shown that both the clustering component and disagreement component of ALFNET are essential for the performance gains.

Chapter 6

Conclusions and Future Work

In this thesis, I discussed the general problem of cost-sensitive information acquisition for structured domains and introduced three techniques for feature and label acquisition. Here, I first summarize our contributions. Then, I discuss potential avenues for future work and then conclude.

6.1 Summary of Contributions

In this thesis, I have discussed feature and label acquisition techniques for classification in relational and non-relational data. Specifically, I introduced a data structure called the *Value of Information Lattice* (`VOILA`) that

- utilized the conditional independence relations that hold in the underlying probability distribution of the features and the labels to reduce the number of possible feature sets to consider for acquisition, and
- allowed efficient sharing of probabilistic inference computations between different candidate sets.

I empirically showed that this technique led to a huge reduction in the space of all feasible subsets of features and that it was able to share computations effectively, making reasoning with sets of features tractable in practice. Through searching

sets of features, I also showed that the most common approach, the greedy policy, often prematurely stopped feature acquisition, judging single features as useless and unable to forecast the benefit of combining multiple features.

I also described two label acquisition techniques for classification in relational data. The first technique I discussed aimed at label acquisition during inference for relational data, assuming a relational model of the domain already exists. I introduced a method named *Reflect and Correct* (`RAC`) that can learn and predict which instances the underlying relational model is likely to misclassify. `RAC` then recommends acquiring the labels of the instances that lie in a high-density misclassified region, maximizing the utility of each acquired label. I showed on both synthetic and real-world datasets that `RAC` significantly outperformed several methods that are based on network structural measures and viral marketing.

The second label acquisition technique I described aimed at constructing training data to learn a relational model. I proposed a method named *Active Learning for Networked Data* (`ALFNET`) that utilizes the explicit links in a network of instances both to select which labels to acquire and to learn a relational model using semi-supervision. `ALFNET` first clusters the network of entities into a small number of groups using the links in the network, and uses the cluster structure to distribute the labels in the network. Then, `ALFNET` iteratively i) uses the existing labels (a few labels might be chosen at random at the first iteration) to learn a relational and a non-relational model, ii) scores each cluster based on the disagreement of the relational model, the non-relational model, and the existing labels in the cluster, iii) picks the clusters where the disagreement is the highest, and iv) acquires the labels

of the most disagreed instance in each chosen cluster. I showed that ALFNET significantly outperformed uncertainty sampling on two real-world publication datasets.

6.2 Future Directions

There are many interesting avenues for future work on cost-sensitive information acquisition. I describe four possible directions here.

6.2.1 Complex Cost Structure

We assumed in our feature acquisition work that both the feature costs and misclassification costs had the same unit and scale and they were monetary costs. However, in practice, there are many additional considerations in addition to minimizing monetary costs and they all need to be optimized simultaneously. For example, the additional considerations for the medical domain might include the following:

- Patients want to receive the necessary treatment as soon as possible, minimizing the number of visits to the doctor and the labs, and minimizing the waiting time for lab test results.
- Some lab tests can be risky for certain groups of patients, thus such tests need to be avoided unless they are deemed to be worth the risk.
- Patients often have preferences for or against certain procedures.

- Patients might ask for immediate medication, such as pain relievers, to reduce the discomfort caused by the problem they are facing, while still waiting for the lab test results. Prescribed medications might provide positive results or might have undesired side effects, changing the status of the patient, and thus requiring different feature acquisitions after the medication is used.

Practical feature acquisition techniques need to take these considerations into account, and minimize monetary cost, number of visits, time to treatment, risk, and patient discomfort, while satisfying the patient preferences as much as possible simultaneously. Formulating feature acquisition as a multiple criteria optimization problem and utilizing and adapting existing techniques (Keeney and Raiffa, 1993; Steuer, 1986) is a promising future direction.

6.2.2 Cost-sensitive Large-scale Data Mining

Recent advances in hardware and software have made it possible to capture large volumes of data. Examples include financial transactions, server logs, news stories, surveillance videos, sensor network data, and tweets. The sheer size of the data often makes it challenging to mine it effectively and efficiently. Timely and intelligent analysis of the data requires fast, accurate, and large scale data mining techniques for sure, but in addition to the fully automated techniques, expert advice and intervention is also needed for at least three cases:

- Even though there is huge amount of data available, we are generally interested in predicting what is not available. Sufficient annotation to learn a (semi-)

supervised model is often unavailable for most domains and tasks. In these cases, expert knowledge is needed to annotate a subset of the instances to construct training data.

- The state-of-the art data mining techniques might not have the desired predictive power for certain domains and tasks. Thus, it might be desirable to consult an expert for difficult instances rather than purely relying on the data mining technique that is being used.

- The domain and task might require making critical decisions that are preferred not to be fully automated. In these cases, an expert might oversee the predictions and intervene if necessary.

It is important in these scenarios to automatically direct the human attention to the right portions of the data, as it will be impossible (or prohibitively expensive) to manually inspect and analyze all the data in a timely fashion. Thus, a promising future direction is to develop large-scale data mining techniques that can automatically determine where human input will be most useful and intelligently integrate the human input into the prediction process.

6.2.3 Visualization Support

Due to its interactive nature, cost-sensitive information gathering can benefit greatly from decision support tools that have strong visual components. Such tools need to be able to:

- Provide a visual and aggregate overview of the predictions of the underlying model, based on what the model currently knows.
- Highlight the pieces of information that are requested from the user and visualize them in relation to the remaining ones.
- Provide visual and analytical reasons why a particular piece of information is requested from the user.
- Allow rich interactions such as letting the user navigate through the instances, change predictions, and provide constraints on the predictions.

We have previously developed an interactive tool named D-Dupe for the task of determining and merging duplicate entries in a database with no primary keys (for e.g., author names and citations in a bibliography database) (Bilgic et al., 2006). The visualization component of D-Dupe has shown to be an effective way of supporting human decisions, as users were able to perform faster, more accurately, and more confidently with the help of D-Dupe compared to tabular style visualization that listed the possible duplicates (Kang et al., 2008). However, D-Dupe did not utilize the human input to revise its predictions for the portions of the database that has not yet been inspected by the user. Developing D-Dupe like tools that can effectively direct the human attention to the right portions of the inference results, while also intelligently utilizing the human input for refining the predictions, is a promising future direction.

6.2.4 Other Prediction Tasks

In this thesis, I considered information acquisition for only classification. Information acquisition for other prediction tasks such as regression and ranking is still under-researched, even though these tasks are also quite common and useful in practice. For example, ranking algorithms are used in every day tools, such as searching the web and providing recommendations to the users. Competitive ranking models such as SVMRank (Joachims, 2002) and LambdaRank (Burges et al., 2006) are trained using data that consist of documents whose relevancy for a given query is typically judged by human annotators. Developing cost-sensitive information acquisition techniques for ranking and regression is still its infancy and thus promises to be a viable future direction.

6.3 Conclusion

In this thesis, I introduced three general techniques for feature and label acquisition for classification. I described how to make reasoning with sets of features tractable in practice for feature acquisition, how to perform label acquisition through a meta-classifier that can learn and predict which instances the primary classifier is likely to misclassify, and how to exploit the underlying network structure to construct effective training data for relational models. With the rising amount of data that is being collected every day and increased adoption of machine learning algorithms in everyday tools and electronics, the importance of pulling human input

and feedback only when necessary and intelligently utilizing it to revise both the underlying model and its predictions will become only more important.

Bibliography

Naoki Abe and Hiroshi Mamitsuka. Query learning strategies using boosting and bagging. In *International Conference on Machine Learning*, pages 1–9, 1998.

B. Anderson and A. Moore. Active learning for hidden Markov models: Objective functions and algorithms. In *International Conference on Machine Learning*, 2005.

Dana Angluin. Queries revisited. In *International Conference on Algorithmic Learning Theory*, pages 12–31, 2001.

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

J. Baldridge and M. Osborne. Active learning and the total cost of annotation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 9–16, 2004.

Valentina Bayer-Zubek. Learning diagnostic policies from examples by systematic search. In *Annual Conference on Uncertainty in Artificial Intelligence*, 2004.

Mustafa Bilgic and Lise Getoor. VOILA: Efficient feature-value acquisition for classification. In *AAAI Conference on Artificial Intelligence*, pages 1225–1230, 2007.

Mustafa Bilgic and Lise Getoor. Active inference for collective classification. In *Twenty-Fourth Conference on Artificial Intelligence (AAAI NECTAR Track)*, 2010.

Mustafa Bilgic and Lise Getoor. Effective label acquisition for collective classification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 43–51, 2008.

Mustafa Bilgic and Lise Getoor. Reflect and correct: A misclassification prediction approach to active inference. *ACM Transactions on Knowledge Discovery from Data*, 3(4):1–32, 2009.

Mustafa Bilgic, Louis Licamele, Lise Getoor, and Ben Shneiderman. D-Dupe: An interactive tool for entity resolution in social networks. In *IEEE Visual Analytics Science and Technology (VAST)*, 2006.

Mustafa Bilgic, Lilyana Mihalkova, and Lise Getoor. Active learning for networked data. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010.

Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *International Conference on Machine Learning*, pages 19–26, 2001.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In *Proceedings of the Neural Information Processing Systems (NIPS)*, pages 193–200, 2006.

Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *ACM SIGMOD International Conference on Management of Data*, pages 307–318, 1998.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

David A. Cohn. Minimizing statistical bias with queries. In *Advances in Neural Information Processing Systems*, pages 417–423, 1997.

David A. Cohn. Neural network exploration using optimal experiment design. *Neural Networks*, 9(6):1071–1083, 1996.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

A. Culotta and A. McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI Conference on Artificial Intelligence*, 2005.

S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *International Conference on Machine Learning*, 2008.

Søren Dittmer and Finn Jensen. Myopic value of information in influence diagrams. In *Annual Conference on Uncertainty in Artificial Intelligence*, pages 142–149, 1997.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1979.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Linda van der Gaag and Maria Wessels. Selective evidence gathering for diagnostic belief networks. *AISB Quarterly*, (86):23–34, 1993.

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3:679–707, 2002.

Lise Getoor, Eran Segal, Benjamin Taskar, and Daphne Koller. Probabilistic models of text and link structure for hypertext classification. In *IJCAI Workshop on Text Learning: Beyond Supervision*, pages 24–29, 2001.

C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. CiteSeer: An automatic citation indexing system. In *ACM Conference on Digital Libraries*, pages 89–98, 1998.

Walter R. Gilks, Sylvia Richardson, and David J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Interdisciplinary Statistics. Chapman & Hall/CRC, 1996.

J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1):122–128, 1986.

Russell Greiner, Adam J. Grove, and Dan Roth. Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139(2):137–174, 2002.

David Heckerman, Eric Horvitz, and Blackford Middleton. An approximate nonmyopic computation for value of information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):292–298, 1993.

David Heckerman, John S. Breese, and Koos Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, 38(3):49–57, 1995.

R. A. Howard and J. E. Matheson. *Readings on the Principles and Applications of Decision Analysis*, chapter Influence Diagrams. Strategic Decision Group, 1984.

Ronald A. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, 2(1):22–26, 1966.

L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-Complete. *Information Processing Letters*, 5(1):15–17, 1976.

David Jensen, Jennifer Neville, and Brian Gallagher. Why collective inference improves relational classification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 593–598, 2004.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142, 2002.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

Hyunmo Kang, Lise Getoor, Ben Shneiderman, Mustafa Bilgic, and Louis Licamele. Interactive entity resolution in relational data: A visual analytic tool and its evaluation. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 14(5):999–1014, 2008.

Ralph L Keeney and Howard Raiffa. *Decisions with multiple objectives: preferences and value tradeoffs*. Cambridge University Press, 1993.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

Andreas Krause and Carlos Guestrin. Optimal nonmyopic value of information in graphical models - efficient algorithms and theoretical limits. In *International Joint Conference on Artificial Intelligence*, pages 1339–1345, 2005a.

Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Annual Conference on Uncertainty in Artificial Intelligence*, 2005b.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, pages 282–289, 2001.

Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web*, 1(1):5, 2007a.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1):1–40, 2007b.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

D. V. Lindley. On a measure of the information provided by an experiment. *Annals of Mathematical Statistics*, 27:986–1005, 1956.

Qing Lu and Lise Getoor. Link based classification. In *International Conference on Machine Learning*, pages 496–503, 2003a.

Qing Lu and Lise Getoor. Link-based classification using labeled and unlabeled data. In *ICML Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003b.

S. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8:935–983, 2007.

S. Macskassy and F. Provost. A simple relational classifier. In *ACM Workshop on Multi-Relational Data Mining*, 2003.

Andrew McCallum and Kamal Nigam. Employing EM and pool-based active learning for text classification. In *International Conference on Machine Learning*, pages 350–358, 1998.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

Luke McDowell, Kalyan Moy Gupta, and David W. Aha. Cautious inference in collective classification. In *AAAI Conference on Artificial Intelligence*, pages 596–601, 2007.

Prem Melville and Raymond J. Mooney. Diverse ensembles for active learning. In *International Conference on Machine Learning*, pages 584–591, 2004.

Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsI. *Mathematical Programming*, 14(1): 265–294, 1978.

Jennifer Neville and David Jensen. Iterative classification in relational data. In *SRL Workshop in AAAI*, 2000.

M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23): 8577–8582, 2006.

M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126, 2003.

Hieu T. Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *International Conference on Machine Learning*, 2004.

Marlon Núñez. The use of background knowledge in decision tree induction. *Machine Learning*, 6(3):231–250, 1991.

Ontario Ministry of Health. Schedule of benefits: Physician services under the health insurance act, October 1992.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, 1988.

Foster Provost, Prem Melville, and Maytal Saar-Tsechansky. Data acquisition and cost-effective predictive modeling: targeting offers for electronic commerce. In *ACM International Conference on Electronic Commerce*, pages 389–398, 2007.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

Matthew Rattigan, Marc Maier, and David Jensen. Exploiting network structure for active inference in collective classification. In *ICDM Workshop on Mining Graphs and Complex Structures*, pages 429–434, 2007a.

Matthew Rattigan, Marc Maier, and David Jensen. Exploiting network structure for active inference in collective classification. In *ICDM Workshop on Mining Graphs and Complex Structures*, pages 429–434, 2007b.

Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 61–70, 2002.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

D. Roth and K. Small. Margin-based active learning for structured output spaces. In *European Conference on Machine Learning*, 2006.

Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *International Conference on Machine Learning*, pages 441–448, 2001.

Donald B Rubin. *Multiple imputation for nonresponse in surveys*. John Wiley, 1987.

Maytal Saar-Tsechansky and Foster Provost. Active sampling for class probability estimation and ranking. *Machine Learning*, 54(2):153–178, 2004.

Maytal Saar-Tsechansky, Prem Melville, and Foster Provost. Active feature-value acquisition. *Management Science*, 55(4):664–684, 2009.

T. Scheffer, S. Wrobel, B. Popov, D. Ognianov, C. Decomain, and S. Hoche. Learning hidden Markov models for information extraction actively from partially labeled text. *Künstliche Intelligenz*, 2, 2002.

Andrew I. Schein and Lyle H. Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265, 2007.

P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, 2008.

Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Neural Information Processing Systems*, pages 1289–1296, 2008.

Burr Settles, Kevin Small, and Katrin Tomanek, editors. *Proceedings of the Active Learning for NLP (ALNLP) Workshop at NAACL-HLT*, 2010.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *ACM Annual Workshop on Computational Learning Theory*, pages 287–294, 1992.

Claude Elwood Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423,623–656, 1948.

Ralph E. Steuer. *Multiple Criteria Optimization: Theory, Computations, and Application.* John Wiley & Sons Inc, 1986.

Ming Tan. CSL: A cost-sensitive learning system for sensing and grasping objects. In *IEEE International Conference on Robotics and Automation*, 1990.

B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Annual Conference on Uncertainty in Artificial Intelligence*, pages 485–492, 2002.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

Peter Turney. Types of cost in inductive concept learning. In *In Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*, pages 15–21, 2000.

Peter D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.

Rongjing Xiang and Jennifer Neville. Pseudolikelihood EM for within-network relational learning. In *IEEE International Conference on Data Mining*, pages 1103–1108, 2008.

Qiang Yang, Charles Ling, Xiaoyong Chai, and Rong Pan. Test-cost sensitive classification on data with missing values. *IEEE Transactions on Knowledge and Data Engineering*, 18(5):626–638, 2006.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Neural Information Processing Systems*, pages 689–695, 2000.

Bianca Zadrozny and Charles Elkan. Learning and making decisions when costs and probabilities are both unknown. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 204–213, 2001.

Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003.