Learning and exploiting context dependencies for robust recommendations

Yap, Ghim Eng

2008

Yap, G. E. (2008). Learning and exploiting context dependencies for robust recommendations. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/41737>

<https://doi.org/10.32657/10356/41737>

LEARNING AND EXPLOITING CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

YAP GHIM ENG

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2008

Acknowledgements

I would like to express my deepest gratitude to my thesis supervisors, Associate Professor Tan Ah Hwee and Associate Professor Pang Hwee Hwa, for their friendly and patient guidance. Throughout my formative years, they have always provided me with frank and prompt advices on my research project. They have allowed me the freedom and time to explore new ideas, and have taught me the discipline and skills to manage and carry out my own work. This thesis and all that I have achieved in this dissertation research could not have been possible without them.

My study is sponsored by a graduate scholarship from the Agency for Science, Technology and Research (A*Star), Singapore. In the past few years, the Management and scholarship officers of A*Star have always been very helpful and receptive to my requests. I have also received a lot of assistance directly from the staff in the General and Graduate Offices of the School of Computer Engineering, Nanyang Technological University. They have always treated me with great kindness and responded quickly and personally to meet all my administrative needs.

I would like to acknowledge the support from my family members and my girlfriend, who have accepted me for who I am and stood firmly by my side all this while. I thank my parents, who have never questioned my choice of career, and have remained totally supportive of whatever I set out to do, tirelessly taking care of my everyday life so that I can focus on my studies. I thank especially my girlfriend of nine years, who is my motivation to work hard in life. She has given me the support and encouragement I need to pursue my dreams. Thank you all.

Table of Contents

Acknowledgements	i
Table of Contents	ii
List of Figures	vi
List of Tables	viii
Abstract	x
1 Introduction	1
1.1 Background and Motivation	2
1.2 Research Scope	3
1.3 Thesis Objectives	4
1.4 Work and Contributions	5
1.5 Thesis Organization	8
2 Literature Review	10
2.1 Context-Aware Computing	10
2.1.1 Context Categories	12
2.1.2 Context Acquisition	12
2.1.3 Context Representation	15
2.1.4 Context-Sensitive Learning	17
2.1.5 Context-Aware Applications	18
2.2 Bayesian Networks	23
2.2.1 Bayesian Network Representation	23
2.2.2 Reasoning within Bayesian Networks	25
2.2.3 Bayesian Networks in Context-Aware Systems	28
2.3 Recommender Systems	29
2.3.1 Collaborative Filtering Systems	29
2.3.2 Content-Based Filtering Systems	30
2.3.3 Context-Awareness in Recommenders	30

- 2.3.4 Bayesian Networks for Recommendations 32
- 2.4 Chapter Summary 33
- 3 Learning the Minimal Set of Contexts and Exploiting the Context Dependencies for Robust Recommendations 34**
 - 3.1 Preliminaries 35
 - 3.1.1 Context-Aware Content-Based Recommender Systems 35
 - 3.1.2 Learning Bayesian Networks from Data with CaMML 37
 - 3.2 Learning the Minimal Set of Contexts 39
 - 3.2.1 A Context Model for Effective Learning 39
 - 3.2.2 Identifying and Learning on the Optimal Set of Predictors 41
 - 3.2.3 Performing Predictions using Learned Bayesian Network 42
 - 3.3 Handling Missing and Erroneous Context Inputs for Robust Prediction 43
 - 3.3.1 Handling Missing Context Inputs 43
 - 3.3.2 Handling a Single Error Context 43
 - 3.3.2.1 Identifying Error Context and Estimating Error Rate 45
 - 3.3.2.2 Cleansing the Training Set 46
 - 3.3.2.3 Predicting on Score using Likelihood Findings to Account for Uncertainty in Context 46
 - 3.3.3 Handling Multiple Erroneous Contexts 48
 - 3.3.3.1 Identifying Error Contexts and Estimating Error Rates 49
 - 3.3.3.2 Cleansing Training Set by Removing Examples with Errors in One or More Contexts 49
 - 3.4 A Restaurant Recommender Application 50
 - 3.5 Experimental Validation 54
 - 3.5.1 Learning the Minimal Set of Important Parameters 54
 - 3.5.2 Prediction with Missing Context Values 56
 - 3.5.3 Prediction with a Single Erroneous Context 60
 - 3.5.4 Performance under Small Number of Training Examples 61
 - 3.5.5 Prediction with Multiple Erroneous Contexts 65
 - 3.6 Further Validation: Personalized Web Page Recommendation 68
 - 3.7 Chapter Summary 72

4	Explaining Recommendations in Bayesian Networks	73
4.1	Preliminaries and Related Work	74
4.1.1	Background of Our Work	74
4.1.2	Bayesian Network and Variable Independencies	75
4.1.3	Related Work on Explaining BN Inferences	78
4.2	Our Approach to Explaining BN Inferences	79
4.2.1	Transforming the BN through Arc Reversals	80
4.2.2	Learning Context-Specific Independencies among Markov Nodes	81
4.2.3	Generating Context-Specific Explanations using Markov Nodes	82
4.3	Explaining BN Compensations during Inferences	83
4.3.1	Explaining BN Compensation of Missing Inputs	84
4.3.2	Explaining BN Compensation of Erroneous Inputs	85
4.3.3	Discussion	86
4.4	Experimental Validation	87
4.4.1	Methods and Procedures	87
4.4.2	Observations under Missing Inputs	90
4.4.3	Observations under Erroneous Inputs	92
4.4.4	Explanation Efficacy	95
4.5	Chapter Summary	97
5	An Effective Knowledge Discovery Process for Noisy Bio-Data Classifications	98
5.1	Introduction	98
5.2	The Problem Domains	100
5.2.1	HIV-1 Drug Resistance Prediction	100
5.2.2	Ovarian Cancer Detection	101
5.2.3	Acute Leukemia Subtype Classification	102
5.3	The BN-Based Knowledge Discovery Process	103
5.3.1	Step 1 - Select Useful Features	103
5.3.2	Step 2 - Capture Feature Dependencies	104
5.3.3	Step 3 - Estimate Error Rates of Features	104
5.3.4	Step 4 - Predict with the Noisy Features	105
5.3.4.1	Likelihoods Estimation	105
5.3.4.2	Explaining the BN Inferences	107
5.4	Experimental Validation	108
5.4.1	HIV-1 Drug Resistance Prediction	108

5.4.2	Ovarian Cancer Detection	111
5.4.3	Acute Leukemia Subtype Classification	114
5.4.3.1	The ALL/AML Data Set	114
5.4.3.2	The Pediatric ALL Data Set	116
5.5	Chapter Summary	117
6	Conclusions	119
6.1	Recap Of Contributions	119
6.2	Limitations and Future Work	120
	References	124
A	List of Publications	139
B	Restaurant Recommender (<i>RR_II</i>)	140
B.1	List of Parameters in <i>RR_II</i>	140
B.1.1	User-Situational Contexts	140
B.1.2	User-Preferential Contexts	140
B.1.3	Restaurant Attributes	141
B.1.4	Restaurant Markers	142
B.2	Decision Tree Representations of Marker Rules	143
B.3	Decision Tree Representations of Causal Models	151

List of Figures

1.1	Thesis Organization.	8
2.1	Typical Architecture of a Context-Aware System.	13
2.2	Context Taxonomy for a Restaurant Recommender System.	17
2.3	Example of a Bayesian Network, With CPTs Shown.	24
2.4	Types of Reasoning, As Illustrated Using the <i>Lung Cancer</i> Example.	26
3.1	The Context-Aware Content-Based Recommendation Process.	37
3.2	A Two-Tiered Context Model.	39
3.3	Procedure for Handling Erroneous Context Inputs.	44
3.4	Context Model of Our Restaurant Recommender Application, <i>RR_II</i>	51
3.5	Causal Dependencies for Markers M3, M11 and M12.	53
3.6	Discovered Bayesian Network in Each Round of Learning.	55
3.7	Average Prediction Accuracy with A Single Context In Error.	62
3.8	Average Prediction Accuracy as the Context Error Rate Varies. Baseline is Decision Tree (DT) Learned on Uncleansed Data.	63
3.9	Average Prediction Accuracy as the Context Error Rate Varies. Baseline is Naive Bayes (NB) Learned on Uncleansed Data.	64
3.10	Average Prediction Accuracy when there are Multiple Error Contexts. Baseline is DT Learned on Uncleansed Data.	67
3.11	Average Prediction Accuracy when there are Multiple Error Contexts. Baseline is NB Learned on Uncleansed Data.	67
3.12	Bayesian Networks Learned from Goats Web Page Data Set for a Real User. . .	69
3.13	Average Prediction Accuracy When There Are Missing Values.	70
3.14	Average Prediction Accuracy When There Are Erroneous Values.	71
4.1	A Bayesian Network (BN) for Movie-Watching.	74
4.2	Example Bayesian Network (BN) and Conditional Probability Table (CPT). . .	76
4.3	Overview of Our Proposed Explanation Procedure, EBI.	80

4.4 Running Example for Section 4.2. Context-Specific Independencies For Target Node E are Captured As Regularities in the Conditional Probabilities of Figure 4.4.c. 80

4.5 Decision Tree (DT) Induced From the CPT Entries for Value e in Figure 4.4.c. . 82

4.6 Example for Explaining BN Compensations. 84

4.7 A BN Learned From the Mushrooms Data. 88

4.8 A BN Learned From the Web Page Data. 88

4.9 The Water BN in Our Experiments. 89

4.10 BN After Reversal of the Arc Coming Out of *Edibility* in Figure 4.7. 90

4.11 BN After Reversal of Arcs Coming Out of *Odor* in Figure 4.7. 90

4.12 Prediction Accuracy With Erroneous Inputs in the Test Data. 93

5.1 Knowledge Discovery from Noisy Data. 103

5.2 Test Prediction Accuracies for TDF. 108

5.3 Error Rate Increments Detected for TDF. 109

5.4 The Initial Error Rates Detected for TDF. 109

5.5 BN Learned From the Ovarian Cancer Training Data. 112

5.6 Markov Blankets in BNs Learned from the ALL/AML Data. 115

List of Tables

3.1	User Preference Rules and Models.	55
3.2	Minimal Set Learning Performance.	57
3.3	Average Prediction Accuracy with Missing Context Values (%). Complete: Data is Complete; Missing: With One Context Missing.	59
3.4	Average Prediction Accuracy for User 1 (%). Error Rate of 30% is Used in All Experiments; Err-Ctx: Context in Error; DT: J4.8; NB: Naive Bayes; BN: Bayesian Network, Using Likelihood Estimates.	61
3.5	User Preference Rule and Model for User 21.	66
3.6	Results of Error Discovery for User 21. Err-Ctx Added: Error Context Added in This Step/Iteration; Err-Ctxs Found: The Error Contexts Found So Far; Err- Rate Ests: Estimated Error Rates for Use in Next Step.	66
3.7	Prediction Performance for User 21. Ctxs_for_Cleansing: Contexts Based On Which Misses Are Removed; Avg. Acc. (%): Average Accuracy on Score; BN: Bayesian Network, Using Likelihood Estimates; DT: J4.8 Decision Tree.	67
3.8	Average Prediction Accuracy of Various Classification Algorithms on the Goats Data Set.	68
4.1	Prediction Accuracy Under Missing Inputs. Complete: All Input Values are Present; Missing: Important Input Values are Missing.	90
4.2	Explanation Generated by EBI for Mushrooms. <i>Odor</i> 's Value is Missing.	91
4.3	Explanation Generated by EBI for Water. <i>CKNI_12_15</i> 's Value is Missing.	91
4.4	Explanation Generated by EBI for Web Page. <i>development</i> 's Value is Missing.	92
4.5	An Example From Mushrooms Data where a Type II Compensation Gives a Different Prediction When <i>Odor</i> is In Error.	94
4.6	An Example From Mushrooms Data Where a Type II Compensation Gives an Identical Prediction When <i>Odor</i> is In Error.	94
4.7	Explanations Generated by EBI for Mushrooms. <i>Odor</i> 's Value is In Error.	95
4.8	Explanation Generated by EBI for Water. <i>CBODD_12_15</i> 's Value is In Error.	96

4.9	Explanation Generated by EBI for Web Page. <i>farm</i> 's Value is In Error.	96
4.10	Average Time (in Units of Seconds) Taken for Generating Each EBI Explanation.	97
4.11	Percentage of Cases With a Reduction in Explanation Complexity After Exploiting CSI.	97
4.12	Explanation Complexity. <i>Before CSI</i> : Average Number of Antecedents Before Exploiting CSI; <i>After CSI</i> : Average Number of Antecedents After Exploiting CSI.	97
5.1	Frequency of Mutations in the Markov Blankets.	111
5.2	The Top 10 Proteins Selected in Decreasing Order of Information Gain.	111
5.3	The Discovered Error Information on the Proteins. The Proteins are Listed in Decreasing Order of Estimated Error Rate.	112
5.4	Examples that are Correctly Classified Only After Applying Our Noise Correction Procedure. The Proteins are Listed in the Same Order As Shown in Table 5.2.	113
5.5	EBI Explanations for the First Test Example in Table 5.4.	114
5.6	Test Accuracy in the ALL/AML Data Set.	115
5.7	EBI Explanations for an ALL/AML Test Example.	115
5.8	Test Accuracy in the Pediatric ALL Data Set. The Best Results for Each of the Six Subtypes are Indicated in Bold.	116
5.9	EBI Explanations for a TEL-AML1 Test Example.	117

Abstract

We consider the recommendation problem, where a set of available items or choices are rated and recommended to users accordingly. Over and above the ratings information used in traditional filtering algorithms, the context of the user-recommender interaction is used to improve the recommendation quality. Specifically, we study how the effective learning and exploitation of context dependencies can help to generate more personal and relevant recommendations.

The use of context in building recommendation systems poses a number of key challenges. Firstly, constraints in system resources make it necessary for recommenders to minimize their context acquisition. While existing recommenders generally adopt an universal set of contexts for all users, a smaller subset of the contexts is often sufficient for particular users. Secondly, the inherent uncertainties in the acquired context values make robustness against missing and error inputs a major concern. Current recommender systems cannot compensate for such noisy context inputs and this can significantly reduce the accuracy of their recommendations. Thirdly, the system recommendations must be properly explained for greater user acceptance. Although numerous explanation approaches are available, none of the existing approaches can effectively account for the context interactions or the noise compensations leading to the recommendation.

In this thesis, we present a set of novel solutions that are designed to effectively address the aforementioned challenges. Our proposed solutions leverage upon the probabilistic inference framework of Bayesian network (BN), which models each context element as a network node, and captures the context dependencies in the form of directed arcs and conditional probabilities.

To minimize context acquisition, we introduce a minimal set learning procedure that iteratively prunes away those nodes which are not directly connected to the target node in the BN that is learned from observation data. The contexts corresponding to the remaining nodes form the minimal context set for that user. In experiments conducted on a restaurant recommendation problem involving strong context dependencies, results show that the proposed procedure reliably identifies the unique minimal set of important contexts for each of the specific users.

To handle noisy context inputs for accurate prediction, we introduce a noise compensation procedure that reliably estimates the contexts' error probabilities by exploiting their learned

dependencies to predict each context in turn from the remaining contexts. The error probabilities are then used to formulate uncertain context values as likelihoods during inference, thereby assisting the BN to compensate for any incorrect context values. We validate the procedure on our restaurant recommender and a web page recommendation problem with only weak dependencies among word attributes. In both problems, our proposed procedure effectively outperforms the J4.8 decision tree and naive bayes classifiers in overcoming noisy context inputs, to generate significantly more robust recommendations for all the users.

To improve user acceptance of the system recommendations, we propose a BN explanation procedure named Explaining BN Inferences (EBI). The EBI procedure automatically explains each inferred target value in terms of the feature interactions within the target node's Markov blanket, which comprises the target node's parents, children and spouses. The process is then recursively applied to explain each Markov value that is missing or corrected during the inference. We evaluate the EBI procedure's efficacy on mushroom classification, water purification and web page recommendation problems. In all the problems, EBI generates high quality, concise and comprehensible explanations for the BN inferences, in particular the underlying noise compensations that enable BN to outperform alternative classifiers, such as decision trees.

Furthermore, in view of the current prevalence of noise in biological data, we consolidate our noise correction and EBI explanation procedures as part of a coherent knowledge discovery process for bio-data classifications. We validate the process on problems including HIV-1 drug resistance prediction, ovarian cancer detection, and acute leukemia subtype classification. By accounting for the uncertainties in the genotypic features' values, our process allows incorrect values to be automatically corrected during classifications, which translates to higher classification accuracy. In addition, the BN recommendations and their underlying noise corrections are succinctly explained by the EBI procedure. The proposed process thus provides an effective analytical tool that can be readily applied to noisy recommendation problems in general.

Chapter 1

Introduction

With the mass proliferation of mobile and Internet access, applications that aim to provide the right information, product, and service to the right person at the right time are flourishing. The individual user would have his or her specific goals, preferences and needs, but the ease of accessing large number of options brings with it the hassle of dealing with choices that are overwhelming or confusing at least. By highlighting the most suitable items to each user, *recommender systems* are fast constituting an indispensable tool for assisting personal decisions.

To use a simple example to illustrate the issues that will be addressed in this thesis, let us imagine a movie recommendation system (e.g. Netflix) that recommends movies to visitors of a movie rental company's web site. Without the recommendations, a visitor without any particular movie in mind could be at a loss when browsing through the tens of thousands of movies available. Now, suppose that in addition to the visitors' past ratings profile, the recommender uses the contents of previously rented movies (e.g. genre, casts, director, scriptwriter, title, synopsis, reviews, songs, pace, scenes, etc) and the contexts of the visitor (e.g. person(s) watching the movie, mood, occasion, location of watching, mode of watching, time of watching, etc) in searching for new movies to recommend. It is obvious that not all of these information is important to every visitor - using all information blindly would lead to non-optimal recommendations in addition to waste of resources (e.g. efforts spent looking for and analyzing hundreds of movie reviews would be wasted if the reviews are not important to the targeted visitor) - so some intelligent processing is needed to identify the minimal set of information important to a specific visitor. Also, some of the information might be missing or in error, and it is important for the recommendation quality to be maintained despite the imperfect information. For ex-

ample, information about the mood of the visitor might be missing during a particular session of interaction, but if the system knows other information such as the occasion (e.g. a date), it should be able to mitigate the effects of the missing information. Last but not least, a visitor would be much more convinced to rent a recommended movie if the reason for recommendation is comprehensible to him. For example, a visitor may be informed that a newly-released romance comedy that is highly-rated by magazines is available, and persuaded to rent the movie as it is his weekend date with his girlfriend and they will be watching the movie alone at home.

1.1 Background and Motivation

A *recommender system* [1], or a *recommender* for short, refers to any general application that is designed to rank a set of available item choices with respect to certain criteria pertaining to its user. For instance, in the information retrieval applications or search engines, the criteria are submitted as queries to the system and the most relevant documents or files are recommended.

Recommender systems are being actively researched due to their obvious commercial values (e.g. [2–6]), and have contributed in no small part to famous e-commerce successes such as Amazon.com. In particular, recent years have witnessed a growing interest in a new generation of recommender systems that are *context-aware* (e.g. [7–10]). With respect to recommenders, the definition of *context* refers to any relevant and potentially useful information regarding to the system users, the available items, and the environment or circumstances of their recommendations. *Context-awareness* then refers to the system’s ability to exploit contexts for improved recommendations. For example, in the task of recommending which advertisements should be placed on a web page that is displayed to a certain user (as described in [10]), where the goal is to select advertisements that would yield high click rates, the contents of the page as well as information about the visitor and the available advertisements are all part of the task’s *context*.

Over the last fifteen years, significant applications of context-awareness have emerged in many different fields of computing including, but not limited to, ubiquitous/pervasive computing, wearable computing, information retrieval and intelligent user interface (e.g. [11–15]). In particular, context-aware recommenders promise wide and viable business applications for the area of mobile e-commerce or *m-commerce* (e.g. [5, 7, 8]). Examples of possible applications

would include downloading promotional coupons onto the handsets of potential customers who are passing by a merchant's display, or recommending someone to visit a Starbucks cafe that is located round the corner instead of travelling more than a mile to the cheapest coffee house.

The inherent fluidity and dynamism of the user-application interactions, especially within the mobile setting, demands that the recommender systems maintain a keen sense of their *context*, or any relevant information that can assist in improving their recommendations. However, mobile context-aware recommenders face many challenges due to their tightly-constrained and volatile operating environment, including the need to minimize context acquisition and to overcome noisy context inputs. Before the recommenders can improve user experience, they must first overcome such issues on the use of contexts in providing pertinent, user-centered advice.

1.2 Research Scope

There are a number of challenging issues that affect most general recommender systems, especially so for context-aware recommenders operating in the mobile environment, and addressing these challenges defines our research scope. We explain each of these challenges in this section.

A first challenge is to identify the *minimal* set of important context parameters for a user. Mobile context-aware recommenders face serious limitations in resources, so they have to conserve their resources by acquiring just the important contexts. In [9], a context is considered important if the average user rating differs across its values, e.g. weather is important if ratings are generally lower when it rains. [8] allows users to explicate their own preferences via rules like “*recommend me a cafe based on its location, but not prices*”, while our earlier work [16] automates the personalization by using *support vector machines* to identify important context for the user. All these prior approaches recognize and exploit the fact that to individual users certain contexts may be more important than others, so they weigh the contexts relative to one another. However, they assume that the causal dependencies between contexts are irrelevant to the prediction of users' decisions, and fail to include these dependencies in the learned model.

The second challenge is the prevalence of *noise* in the context inputs due to missing and erroneous values. Within the volatile mobile environment, lapses in the context acquisition occur frequently due to unforeseeable circumstances like failure to negotiate access rights to

CHAPTER 1. INTRODUCTION

protected data, faulty sensors, and intermittently broken or unstable communication links. Similarly, many sources of context information including hardware and software sensors are prone to errors. Context-aware recommenders have to be robust so that they can maintain accurate predictions even under such imperfect conditions. To the best of our knowledge, no prior work has considered exploiting the dependencies among contexts to address noisy inputs. In particular, learning methods like *decision trees* and *support vector machines* do not encode context dependencies, and thus cannot compensate readily for missing and erroneous context inputs.

The third challenge lies in getting the user to understand and accept the recommendations. Ultimately, the recommender systems are meant to advise users on complex decisions, and user acceptance is more likely if the reason for the given advice is clear and understandable. As the domain becomes increasingly complex, the gap between the user's and the system's knowledge will widen, and the need for explanation will increase. Since each of the system's recommendations is usually a product of the interactions among multiple contexts, the explanations for the resulting recommendations should reflect the relevant context interactions. Furthermore, where missing or erroneous context inputs are compensated during the recommendation process, the accompanying explanations should be able to account for the underlying noise compensations.

1.3 Thesis Objectives

In this thesis, we aim to address the aforementioned challenges faced by context-aware recommender systems. The specific goals of this dissertation research are summarized below:

- Each individual is unique, and so should the system recommendations to her. In particular, we note that users place different emphasis on different context elements for specific recommendation task, and it should be possible to automatically identify the minimal set of important interacting contexts that is relevant to each user.
- Noisy inputs including missing and erroneous context values can reduce the accuracy of recommendations. We observe that in many domains, dependencies exist between the context features in the data, and it should be possible to exploit these input redundancies to automatically detect as well as compensate for any noisy inputs.

- In addition to predictive accuracy, the ability of the system to explain the resulting recommendations is important for greater user acceptance. Since the recommendations are based on the input values, it should be possible to automatically generate rules that concisely explain the system output in terms of a salient set of input values.

To realize these objectives, our research champions the effective learning and exploitation of *context dependencies*, or the statistical correlations between context values. These dependencies are captured in the acyclic graphical model of Bayesian network (BN) [17], which provides a solid statistical framework that represents the contexts as nodes and concisely describes their dependencies in the form of arcs and conditional probabilities.

The techniques that are developed and presented in here are designed to be equally applicable to general recommendation tasks, including that of pattern classification. Since all classifiers including cancer diagnostic systems are instances of recommender systems, where each predictor attribute or variable can be considered as a piece of context information that is relevant and useful to the recommendation task, success of the undertaken research would provide far-reaching benefits to the general computer science community.

1.4 Work and Contributions

Over the past few years, we have developed and validated specific Bayesian network-based technologies that systematically learn and exploit context dependencies, so that not only can a recommender identify the minimal set of interacting contexts for specific users, the dependencies can assist the recommender in compensating missing and erroneous inputs, while also helping to explain the final recommendations for improved user acceptance. The following is a summary of the main research contributions that are presented in this thesis, as well as the publication containing each of these contributions:

- **Learning the Minimal Set of Contexts**

We have devised a Bayesian network (BN)-based recommender that learns the *minimal* set of important contexts for each specific user so as to minimize the context acquisition. The

learning procedure iteratively discards all those context parameters that are not directly connected to the user rating variable within the learned BN. The remaining parameters constitute the minimal context for that user. In applications that reason with defined markers, i.e., higher-level concepts derived from the primitive context parameters, we apply the procedure on the markers. Those context parameters that feed into the minimal marker set constitute the minimal context. Since there are typically far fewer markers than contexts, learning on markers allows a user's correct minimal context to be discovered from fewer examples. Our procedure and the validation results have been published as a full paper in the proceedings of the Seventh International Conference on Mobile Data Management (**MDM 2006**) [18].

- **Handling Missing and Erroneous Context Inputs for Accurate Prediction**

Next, we harness the dependencies among the contexts to make the recommender resilient against missing and erroneous context inputs. We have proposed a two-tiered context model that captures the context dependencies in the learned BN, which enables available contexts to effectively compensate for unspecified or missing input values during the recommendations. In addition, we have invented a novel procedure for overcoming errors in the context inputs. The proposed procedure employs the learned BN to estimate the error probability for each context parameter using just the training examples. If the training sample size is large, the procedure discards those inconsistent or outlier examples, and learns a new network from the cleansed data. During subsequent predictions, the erroneous context inputs are entered as likelihood findings or soft evidence so as to account for their uncertainties.

We have evaluated the above minimal set learning and noise compensation procedures on a restaurant recommendation problem that involves defined markers and strong causal dependencies among contexts. We have also experimented with a web page recommendation problem where there are only weak dependencies between word attributes. In both problems, our iterative learning procedure reliably identifies the minimal set of important contexts for each user. BN also consistently outperforms J4.8 decision tree and naive bayes classifier in overcoming missing and erroneous context inputs, to produce significantly more accurate recommendations. Our procedures and their validation results have been published in the July 2007 issue of the IEEE Transactions on Knowledge and Data Engineering (**IEEE TKDE**) [19].

- **Explaining Recommendations in Bayesian Networks**

To address the remaining challenge of explaining the BN recommendations to the users, we have introduced a procedure named **Explaining BN Inferences (EBI)**. The EBI procedure considers just the target node's Markov values during inference, because conditional independence implies that these values fully explain the inference. To simplify the explanations, EBI exploits context-specific independence relations among the Markov nodes. Besides its unique ability to exploit variable interactions for explaining recommendations, EBI is also unprecedented in its ability to explain relevant noise compensations, which are otherwise hidden during inferences, as incremental layers of explanations that are presented to the interested users upon request.

We have validated the proposed EBI procedure on a variety of problem domains, including mushroom classification, water purification and web page recommendation. The results show that EBI generates high quality, concise and comprehensible explanations for BN inferences, in particular the underlying compensation mechanism that enables the BN to outperform alternative recommender systems, such as the decision tree classifiers. The EBI procedure and its validation results have been published in October 2007 by the International Journal of **Applied Intelligence** (APIN) [20].

- **An Effective Knowledge Discovery Process for Noisy Bio-Data Classifications**

We have introduced an effective knowledge discovery process for noisy bio-data classification, which integrates our noise correction and EBI explanation procedures. We have validated the process on various benchmark bio-classification domains, including HIV-1 drug resistance prediction based on viral mutations, ovarian cancer detection based on protein mass-to-charge ratios, and acute leukemia subtype classification based on gene expression data. By learning a BN from the data, we capture the dependencies among selected genotypic features, which we exploit to estimate each feature's error rate. Taking these uncertainties into account allows noisy input values to be automatically corrected during classifications, leading to higher classification accuracy. Moreover, the EBI procedure generates concise and informative explanations for both the BN recommendations and the underlying noise corrections. Our findings on ovarian cancer detection based on protein mass spectra data have been published in the Twenty-Second National Conference on Artificial Intelligence (**AAAI 2007**), as one of the few papers to be selected for both full oral and poster presentations [21].

Figure 1.1: Thesis Organization.

1.5 Thesis Organization

Figure 1.1 outlines the organization of this thesis. As shown in the figure, the main body of the thesis has been organized into chapters, each covering a distinct topic within our work.

Chapter 2 reviews the major areas that are central to this thesis, by following the sequence of *context-aware computing*, *Bayesian networks*, and *recommender systems*. We have deliberately chosen this order of presentation so that as the different major areas are covered, we are able to gradually bring forth those works that overlap and link up these sections. For instance, Section 2.2 highlights the current applications of Bayesian networks in context-aware computing, and Section 2.3 describes the use of both context-awareness and Bayesian networks in recommendation problems. The literature on the BN explanation methods are deferred to Section 4.1 of Chapter 4 for a clearer presentation.

Chapter 3 presents the first and the second of our research contributions as described in Section 1.4. Firstly, it presents our two-tiered context model for effectively **learning the minimal**

CHAPTER 1. INTRODUCTION

set of contexts, so that the relevant context dependencies are captured in the learned Bayesian network. Secondly, our proposed procedures for **handling missing and erroneous context inputs for accurate predictions** are introduced in details. Following that, the chapter describes a modelled context-aware restaurant recommender and discusses the evaluation results based on this system. Further validation results on a real-world web page recommendation problem are presented before the chapter concludes.

Chapter 4 introduces the third of our research contributions on **explaining recommendations in Bayesian networks**. The chapter first describes the relevant Bayesian network (BN) concepts that are related to variable independencies, and then reviews the existing BN explanation methods. Following that, the chapter introduces our **Explaining BN Inferences (EBI)** procedure for explaining BN recommendations using variable interactions, and discusses how EBI explains the underlying BN compensations for missing and erroneous inputs. Evaluation results on EBI's efficacy for problems including mushrooms edibility prediction, water treatment and web page recommendation are then presented.

Chapter 5 introduces the fourth of our research contributions, **an effective knowledge discovery process for noisy bio-data classifications** that consists of our novel noise correction and EBI explanation procedures. The proposed process is evaluated using multiple real-world bio-classification problems. Specifically, in the HIV-1 drug resistance prediction problem, we verify that the target's Markov nodes in the automatically learned BNs indeed correspond to known drug-resistance mutations. Furthermore, we verify that the proposed process correctly identifies all the features with noise, and that the noise correction significantly improves the classification robustness of BN. Following that, we validate the proposed process on the tasks of ovarian cancer detection based on proteomic expressions from human blood sera, as well as acute leukemia subtype classification based on gene expressions from microarray data sets.

As the final chapter of this thesis, Chapter 6 concludes the dissertation report with a recap on our research contributions, followed by a discussion on some limitations and future work. A list of the publications resulting from this dissertation research is provided in Appendix A.

Chapter 2

Literature Review

We address the aforementioned challenges in context-aware recommenders using Bayesian network, and this review covers the major aspects of our problem: context-aware computing, Bayesian networks, and recommender systems. The sections are connected with references to the overlapping work, for example, a review on the existing context-aware recommenders, etc.

2.1 Context-Aware Computing

Context refers to any information that provides meaning to the situation of interactions between the users and applications; *context-awareness* is then the ability to make sense of such information and use them to improve the interactions as they unfold. This review introduces the reader to context-aware computing, provides a flavor of the major issues in the area, and surveys some of the existing systems. As this thesis focuses on learning and exploiting context dependencies for robust recommendations with explanations, the discussions of important outstanding issues such as context-acquisition and context-representation would only be covered in brief.

The term *context-aware* first appeared in Schilit and Theimer's 1994 paper [22]. We can use the sensor-oriented view on context-awareness by Schmidt and Laerhoven [23] as an example:

Situation The real-world situation;

Sensor data The data captured to represent the situation;

Context The abstract description of the real-world situation; and

Context-aware application The applications (software, hardware, appliances) that change their behavior according to the context [23].

CHAPTER 2. LITERATURE REVIEW

Like the above definition, earliest attempts to define *context* had either suggested general synonyms for context, such as “aspects of the current situation” [24], or “elements of the user’s environment which the computer knows about” [25], or had enumerated examples of contexts, such as *identity*, *environment*, *time* and *location* (e.g. [22, 26, 27]). As the suggested synonyms are somewhat vague while examples tend to be over specific, Dey and Abowd [28] had conjured a definition that aims to assist the application developers in enumerating the relevant or useful contexts for any scenario. Their words are regarded as the *de facto* definition of *context*.

Definition (Context): Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the users and applications themselves [28].

Dey and Abowd defined context-aware applications as applications that respond to different contexts, be it by *actively* adapting their own behaviors or *passively* presenting the context information to their users. They defined *context-awareness* as follows.

Definition (Context-Awareness): A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task [28].

Put simply, the main characteristic of a *context* is that it describes the situation of the user and other entities involved in the fulfillment of a task; a system is context-aware as long as it uses these contexts to improve the task performance. For the recommendation problem which we consider in this thesis, we use “item attributes” to refer to the descriptor features of the objects for which the recommendation is wanted, and “user contexts” to refer to the context parameters that characterize the user (e.g. the user profile, preferences, and the time, location and user activity). The item attributes are specific to each recommendation application whilst many of the user context parameters (e.g. weather, age, gender) are portable across different applications. In a music recommendation application, for instance, user contexts would include a user’s gender and age, while item attributes describing the songs would include e.g. genre and tempo. Clearly, a recommender that is capable of taking into consideration a specific user’s gender and age would be better positioned to provide a more relevant song recommendation.

2.1.1 Context Categories

How the elements of context are categorized directly influences the design of any context-aware framework, because the actual contexts that are used tend to be enumerated from the relevant categories. There exist numerous classification schemes in the literature.

Schilit et al. [29] defined three classes of context specific to mobile applications, namely 1) *computing context* e.g. network status, bandwidth, communication costs, device capabilities and resource (printer, etc) availability, 2) *user context* e.g. location, schedule, profile and activity, and 3) *physical context* e.g. lighting, noise, temperature and weather.

Dey and Abowd [28, 30] named *person* (individuals, groups), *place* (room, building), and *object* (devices, tools) as the entities to consider when dealing with contexts. Each of these entities may then be described by context attributes that belong to four categories: *identity* (an unique identifier), *location* (the entity's position, proximity), *status/activity* (e.g. temperature in a room, processes on a PDA) and *time* (timestamps, order of events).

Koolwaaij and Strating of the WASP ("Web Architectures for Services Platforms") project classified user context into several even finer categories, according to whether they belong to environmental, personal, task-related, social, historical, or spatial-temporal considerations [31]. Their example: *Imagine a traveller in Amsterdam (spatial), who is heading for his hotel (task), likes public transport more than taking a taxi (personal), and travels with his wife and one-year old child (social) while it is a rainy day (environment).*

There appear a number of important consensus that we can gather. Firstly, a user's context is an abstract description of her real-world situation, such that specific contexts can be acquired to influence the applications' behaviors [23]. Secondly, contexts can rarely be obtained from one single source; they usually come from heterogeneous and distributed sources having varying degrees of accessibility and reliability [31]. Thirdly, regardless of how the actual elements may be categorized, it is important to recognize that contexts are very much an inherent and non-separable part of our everyday activities [28, 29, 31, 32].

2.1.2 Context Acquisition

For contexts to be useful, the acquisition process has to take account of their *relevance* to the users and tasks. What might be of great importance to some users could be useless to

Figure 2.1: Typical Architecture of a Context-Aware System.

another, and important contexts for one application could be redundant to many others. There is usually a need to resolve the acquired data (e.g. a physical sensor reading) into information relevant and useful to an application (e.g. surface wetness of leaves in wheat plants). Careful considerations have to be allocated, when acquiring contexts, to both the context sources and the users (humans and computer applications) of the information.

Figure 2.1 shows the typical architecture of a generic context-aware system. Generally, we can expect external context sources to supply the necessary context information, but these information may not be in the form that is acceptable by the context-aware applications. A *context manager* thus serves the important purpose of processing and presenting the relevant contexts in the manner that is expected by the applications. Applications can receive the contexts via a simple request-response process, or they can subscribe to the manager, which would then push the relevant contexts to the individual application whenever some predefined conditions are met. The framework hence provides applications with access to information on contexts which are useful to the human user.

With reference to the classical model of provider-consumer interactions, existing approaches in context acquisition can be broadly classified as either *provider-centric context acquisition* or *consumer-centric context acquisition* - the former centers upon the context providers and caters to new users as they are introduced, whilst the latter focuses on the requirements of consumers (human users and context-aware applications) and sources for contexts to satiate them.

Provider-Centric Context Acquisition

The provider-centric approach to resolving contexts was mainly evident in service-based infrastructures, e.g. WASP [31]. These were source- or provider-centric in that the system clearly

CHAPTER 2. LITERATURE REVIEW

knew the existence, requirements, capabilities, interfaces, and in some cases, the reliability or quality of each and every available sources of contexts. A source of context can be a hardware or software sensor, an external service such as the SMS-based weather and information services that are now available [33, 34], or it can be a locally accessible storage (such as a dataset or a database) with relevant information that is useful for the task, e.g. user preference profile for a movie recommendation task, or gene expressions for disease classification.

Consider an example scenario in which a certain lady is interested in purchasing a particular dress (or a book, or an iPod, etc). Relevant contexts under this scenario include the lady's location, and the prices and availability of the desired item(s) at nearby stores and sites. Services are already available for users to receive such real-time price updates by SMS requests [33, 35], and providers like Google SMS [33] have recognized the benefits from knowing the user's location, which incidentally is the single most extensively explored context data [36, 37].

In fact, a whole myriad of location-based services (LBS) had already been developed to exploit the ease at which mobile phone users could be tracked or accurately located, mainly by mobile network providers, e.g. emergency assistance such as vehicle breakdown and ambulance services. Some governments had legislated that mobile network providers must be able to locate a user in real time with high accuracy, e.g. within fifty meters [36].

Consumer-Centric Context Acquisition

Many existing systems are centered on particular applications or category of applications, e.g. the HP Lab's Cooltown project [38] and the Context Toolkit [39]. The most important consideration for such systems was relevance to the particular application, which could be clearly seen from the way the context data were modelled to satisfy and fulfill the application's specific needs. For example, an itinerary planner would likely require weather or traffic condition forecasts or updated event schedules, etc. For each of these required contexts, the most authoritative source was sought by the API implementer of the application's domain and after some form of negotiation (subscription, registration, etc), the application requirements would be directly bound to these specific sources (a sensor, a group of sensors, a web service, a database, an information repository, etc).

2.1.3 Context Representation

Power [40] argued that there would not be a globally standard model for representing context information. Many current approaches advocated pre-defined context models, which applications interacted with via middleware platforms for querying and manipulation. However, context data came in many different forms, from many different sources. A challenge in ubiquitous computing was thus to take all this information from disparate and heterogeneous sources to construct a useful representation for querying by context-aware applications. This representation must then be kept up to date with changes within the environment. Rather than being managed globally, context information would almost definitely be managed at the organization level, such as a business or even a simple home user, due to scaling privacy and security concerns. Power's paper examined topic maps as a means to addressing the requirements for a Context Information System (CIS). It stressed that the gathering, interpretation, storage and real-time dissemination of context, collectively known as context management, must facilitate the seamless accessibility to context services by applications as the user moves between different environments.

Ranganathan and Campbell [41] proposed a context model based on first order predicate calculus, which allowed for clear specification of the power and expressiveness of the model, i.e., the exact types of contexts that could be expressed, types of rules that could be evaluated, and types of queries that could be handled. The model allowed for distributed reasoning, where each component obtained the contexts it needed and carried out its own reasoning, e.g., via rule-based inferences. A developer specified the actions for different contexts, and the system obtained the context, evaluated the rules and performed the actions. The structures of different context predicates were specified in an ontology written in DAML+OIL, which could be mapped to representation formats like plain text, XML, a tuple in a relational database or a serialized object. Ranganathan and Campbell noted that the model can be enhanced with first order temporal logic, e.g. by expressing the constraint that someone must enter a room before they can leave it.

Gu et al. [42] proposed a formal model of context based on OWL (Web Ontology Language) that addressed issues including semantic context representation, context reasoning,

CHAPTER 2. LITERATURE REVIEW

knowledge sharing, context classification, context dependency and context quality. They defined a common upper ontology to model the four concepts of *person*, *location*, *computational entity* and *activity*, and a set of low-level domain-specific ontologies to define the details of these four concepts in specific domains. A low-level ontology could be dynamically replaced as the environment changed. They defined direct context (including sensed and defined context) as those acquired from a provider directly, and indirect context (including aggregated and deduced context) as those obtained by interpreting direct context via aggregation and reasoning. The model captured relationship between contexts by introducing dependency tags, and captured quality of context by annotating contexts with extensible quality constraints associated with quality parameters of accuracy, resolution, certainty and freshness, e.g. defined context was more reliable than sensed or deduced context, and RFID-based locators more accurate than Bluetooth ones.

In an earlier paper [16], we presented a generic framework for recommender systems that represented contextual information in the form of domain-driven context taxonomies. The context information was coded in extensible markup language (XML), and the structure of the context taxonomy was captured in a document type definition (DTD). In an example restaurant recommender system, we implemented several SOAP services on a J2EE server as external sources for context information including *location*, *weather* and restaurant-related data. Part of the associated context taxonomy is shown in Figure 2.2. Each context is treated as a separate entity that is relevant to the task. Such a modular design makes the applications amenable to future addition and removal of contexts, and promotes efficiency by freeing the application developers to focus on the task at hand.

Additional information on context-modelling could be found in [43], which provided a thorough survey of the most relevant current approaches to modelling context. Although the survey was geared towards the more specific area of ubiquitous computing, many of the analyzed context models are general to context-aware computing. These include key-value models, markup scheme models, graphical models, object-oriented models, logic-based models, and the ontology-based models. The review paper concluded that the most promising class of models for context was the ontology-based models, but stressed that this did not mean that the other approaches were unsuitable for ubiquitous computing.

Figure 2.2: Context Taxonomy for a Restaurant Recommender System.

2.1.4 Context-Sensitive Learning

There exist few prior works in the area of context-sensitive learning. We broadly classify these under the domains of machine learning and sensor fusion. Within machine learning, Turney [44] proposed *contextual weighting* as a new strategy for using context information when classifying, with the aim of preferring some features over others to improve accuracy.

Definition (Contextual weighting): The contextual features can be used to weight the primary features prior to classification. The purpose of weighting is to assign a greater importance to features that, in a given context, are more useful for the classification. Contextual selection of features might be viewed as an extreme form of contextual weighting - the selected features are considered important and the remaining features are ignored.

Within Turney's works, the primary and contextual features are mutually-exclusive [44–46]. Contextual weighting basically involves the assignment of weights to primary features given the values of a set of contextual features, which do not possess any discriminatory power on their own. A closer analogy exists between the fusing of sensor outputs to infer higher-level

CHAPTER 2. LITERATURE REVIEW

contexts with greater confidence, and our objective of identifying the truly relevant contexts for a user so as to achieve a better recommendation performance.

Definition (Sensor fusion): Wu et al. [47] defined the process of *sensor fusion* as the research of intelligently integrating multiple sensors and multiple sensor modalities.

The sensor fusion process combines the outputs of sensors pre-specified to provide like-kind of context information: agreements elevate confidence estimations whereas conflicts suppress the measurement confidence for that corresponding context. Wu et al. [48, 49] applied the Dempster-Shafer Theory (DST) of Evidence as their sensor fusion algorithm. Assuming that sensors could report both the context information and the corresponding confidence estimations, Wu et al. exploited the sensors' predefined interdependencies or relationships in order to infer the higher-level context with greater confidence. Similarly, once we have discovered the underlying interdependencies among the contexts for a particular user, it would then be possible for us to exploit the relative importance amongst the context parameters so as to improve the recommendation accuracy for that user.

A recent work by Guan et al. [50] discussed the selection of low-level sensors for inferring high-level contexts, based on each sensor's information gain contribution towards the accuracy of predicting a given context. Since the information gain contributions of sensors are computed independently and then ranked according to their relative discriminatory power in a reasoning process, dependencies among the sensor outputs are completely ignored. The sensor selection is performed before context reasoning, and the accuracy of the resulting reasoning for different number of sensors and different learning methods is investigated. Their results suggested that minimizing the set of low-level contexts can significantly reduce the computational complexity of the model and the size of the data repository while maintaining good reasoning performance.

2.1.5 Context-Aware Applications

Dey and Abowd [28] suggested that context-aware systems should only *respond* to contexts; the detection and interpretation of contexts should be the responsibilities of other systems. Based on this understanding, they identified three defining functions of context-aware applications, namely *presentation of information and services*, *automatic execution of a service*,

CHAPTER 2. LITERATURE REVIEW

and *tagging of context to information for later retrieval*. These three categories are developed from the taxonomies raised by both Schilit et al. [29] and Pascoe [51], and it appears that most existing systems perform one or more of these three functions.

One of the first recognized context-aware applications is the Active Badge Location System introduced by Want et al. in 1992 [52]. The infrared-based system determined a user's location and forwarded phone calls to the telephone that was closest to that user.

In a pioneering paper, Schilit et al. [29] described PARCTAB, a palm-sized wireless device that acted as a graphics terminal for applications running on remote hosts. A room formed a network cell, and the PARCTAB sent identifying beacons to the base periodically, which then notified the relevant applications. The prototype demonstrated four categories of context-aware applications defined by [29]: *proximate selection*, *automatic reconfiguration*, *contextual information and commands*, and *context-triggered actions*. *Proximate selection* is an interaction technique where nearby objects are emphasized or made easier to choose. *Automatic reconfiguration* involves context-based auto-reconfiguration of system components. *Contextual information and commands* exploits the fact that people performs different tasks in different places, e.g. different files are retrieved depending on the location where the command is issued. *Context-triggered actions* are IF-THEN rules on how a system should adapt to contexts.

Assuming that the information and services that a user accesses strongly depend on his or her location, which could be segregated into cells, the Nexus project [53] suggested the technique of *universal hoarding* to improve user access. Whenever a registered mobile user crossed over a cell-boundary, the info-station in the new cell transferred to that user's mobile device as much information as that user was expected to require during the stay. Mobile applications then exploited the cached data to provide fast services to the user.

Hori and Aizawa [54] advocated the use of contexts to annotate video systems. Their wearable system continuously captured data from a wearable camera and a microphone, and annotated these using contexts extracted from sensors including a brain-wave analyzer, a GPS receiver, an acceleration sensor, and a gyro sensor. Keys for retrieval were then given by motion data, face recognition, time data, weather information, GPS data, and brain-wave data. The audio-visual data could also be annotated and retrieved from large database using keys specified by the user or using combination of the above keys.

CHAPTER 2. LITERATURE REVIEW

Hess and Campbell [12] described a file system that could organize its application data based on context, import the user resources based on physical presence, and also support file format conversions to accommodate device context. Specifically, context information was used to automatically make personal storage available to applications as conditioned by a user's presence, to organize the files to allow easier locating of data for applications and users, and also to retrieve data in a format suitable for the user preferences or device characteristics. A user might merge his or her personal data into the local space by either carrying their own personal data references with them via a hand-held, or authenticate themselves via their fingerprints and retrieve their data references from a profile server.

Hexel et al. [55] presented a computerized system for supporting public presentations that adapted large-screen projected presentation based on who were in the audience. The system also used the personal display devices of individuals in the audience to provide a complementary display with both content and presentation adapted to the individual.

Kern et al. [13] used wearable computers and sensors to generate personal contextual annotations in audio-visual recordings of meetings. They argued that such annotations were essential and effective to allow retrieval of relevant recordings from large audio-visual databases. The paper provided a good introduction to the area of wearable computing.

To integrate physical and virtual worlds into a mixed reality, Romero and Correia [56] presented a *hypermedia* model that abstracted the details of each information item so as to provide a generic interface for information navigation and structuring. Each component had attribute-value labels that defined its characteristics or associated user preferences, and this enabled context searching and adaptive filtering of information based on events.

Bardram [57] presented hospital works as a strong case for context-aware computing by describing their work in designing, developing and evaluating context-aware clinical applications. These included an Electronic Patient Record (EPR), a pill container, and a hospital bed that had an integrated computer with a touch sensitive screen. The bed had RFID sensors for identifying the patient lying in the bed, the clinician standing beside the bed, and various other medical 'stuff' with RFID tags embedded. The paper outlined some key design principles behind a context-aware framework for clinical applications.

CHAPTER 2. LITERATURE REVIEW

Carswell et al. [58] described the MoCHA system that enabled its client mobile devices to display spatially or semantically relevant city cultural heritage. The contexts employed included location, device display characteristics, user preferences and profile (e.g. the user's interests, age, task, etc), as well as bandwidth and protocol capabilities. The JENA Semantic Web Toolkit from HP Labs was used for semantic representations. Information accessed in the past, activities planned by the user, and the situation dependencies (e.g. location) of these activities were all being used to infer future information requirements.

Fujinami et al. [59] presented *ContextDistillery*, a Java application framework that allowed context-aware applications to be developed quickly without taking into account details of sensing. Context was modelled either as *base-context*, including identity, time, location and state (e.g. "is seated"), or as *meta-context*, which gave information about context: quality of context (precision, accuracy, etc) and sensor specification (sensitivity, lifetime, failure probability, unit, data format, poll-able or not, etc). In their experiments, RF-tags were added to an umbrella and a mobile, and both the user schedule (on *Yahoo! Calendar*) and the weather forecast (on *Yahoo! Weather*) were monitored. An email reminder would be sent to the user's mobile phone if rain was predicted and it was sensed that the user was leaving without his umbrella. They assumed that a mobile would always be with its owner and hence represented the user's presence. On the user side, they assumed a PDA with a personal database that held private data, e.g. birthday.

Terrenghi and Zimmermann [15] described the LISTEN project, which used immersive audio to augment users' experience in a museum environment. They focused on issues concerning multi-modal interaction, specifically aural-visual perception principles, and stressed on the importance of augmenting the real visual environment in a personalized way using context modelling techniques. The user's proximity to the displayed item was used, but the authors acknowledged that this could not guarantee the user's real focus, since he or she might actually be looking at another object in another sound zone. The system was further limited by its use of predefined user stereotypes, which could not be altered at runtime, for defining the user's type in order to adapt the scenery accordingly.

Based on the first order predicate context model in [41], the Gaia infrastructure was developed to enable context-awareness in ubiquitous computing environment. Gaia used a provider-

CHAPTER 2. LITERATURE REVIEW

consumer design, where context providers collected contexts for applications to access. Context history was stored and made accessible in a database, allowing data mining to learn rules about user and application behaviors, e.g. for predicting moods and detecting intrusion. The Gaia model used rules and machine learning approaches to infer new contexts from existing contexts, and a priority-based mechanism to resolve context conflicts. Context-aware applications had a configuration file that associated certain contexts with a method that was called when the context became true. The application developer implemented these methods, as well as functional terms that allowed the reasoning engine to learn about the state of the application. To minimize the number of context expression evaluations whenever the context changed, they organized all context expressions in a poset data structure. Communication between entities took place via CORBA, and rules and queries were evaluated using the XSB reasoning engine.

Hatala et al. [60] presented ec(h)o, an “augmented reality interface” for museums that utilized spatialized soundscapes and a semantic web approach to information. It allowed people to interact with the system via movement and object manipulation, without the direct use of a computer device. The retrieval mechanism was built on the user model and conceptual descriptions of digital sound objects and museum artifacts defined as various ontologies. The retrieval criteria were inference rules that represented knowledge from psychoacoustics, cognition domain and composition aspects of interaction. User interests were inferred from user movement through the exhibition and from user interactions with the system. Sound objects were then retrieved based on their relevance to the user.

Interestingly, despite the large number of context-aware applications over the years, Korkea-aho [61] observed that most systems had used only the *time*, *location*, and *identity*. Other useful contexts that were seldom used included social situation (e.g. relationship with the next person), activity (e.g. rehearsing a speech to oneself), and environmental aspects (e.g. whether it was raining and how long the rain was expected to last). Reviews on context-aware applications over the last decade unanimously agreed that *location* was still the single most dominant context being exploited [36, 37, 62]. The recent survey by Baldauf et al. [37] found that such external (or physical) contexts were used by virtually all existing context-aware systems, whilst examples of applications that had attempted to use internal (or logical) contexts like user goals,

activities and preferences were relatively few and far between (e.g. the conference assistant application built by GIT [63], and the IntelliZap system [64], which used a text query marked by the user in a viewed document to initiate a search guided by the text around the query).

2.2 Bayesian Networks

Korb and Nicholson [65] provided the following concise definition of *Bayesian networks*

A Bayesian network (BN) is a graphical structure that allows us to represent and reason about an uncertain domain. The nodes in a BN represent a set of random variables from the domain. A set of directed arcs (or links) connects pairs of nodes, representing the direct dependencies between variables. Assuming discrete variables, the strength of the relationship between variables is quantified by conditional probability distributions associated with each node. The only constraint on the arcs in a BN is that there must not be any directed cycles: you cannot return to a node simply by following directed arcs.

As a probabilistic recommendation system, Bayesian network (BN) advises us on complex decisions. Given a set of input values, known as the *findings* or *evidence*, a BN will derive the posterior probabilities for a target of interest. This is known as an *inference* on the target, and the target value with the highest belief or probability is the *prediction* or the recommendation.

This section begins with a brief introduction to the representation (the *syntax*) of BN, followed by a discussion on the various fundamental processes of reasoning (the *semantics*) in a BN. For these purposes, we reference closely the informative text of Korb and Nicholson [65].

2.2.1 Bayesian Network Representation

The nature of each random variable of interest, known as a *node* of the BN, can either be discrete (where conditional probabilities are presented in a table) or continuous (with conditional probabilities represented as, for example, a Gaussian distribution), but here we consider just discrete nodes as this is necessary and sufficient for most purposes. The possible values, or

CHAPTER 2. LITERATURE REVIEW

Figure 2.3: Example of a Bayesian Network, With CPTs Shown.

states, of a discrete node must be *exhaustive* and *mutually exclusive*, meaning that a discrete node can only take exactly one of its specified values at any time.

The network topology captures the *qualitative* relationships among the network nodes, where two nodes are directly connected by an arc if and only if one of the nodes *affects* or *causes* the other, with the effect's direction being indicated by their connecting arc's direction. We call the causing node the *parent node*, and the affected node the *child node*. By convention, the networks are laid out with the arcs pointing from top to bottom. Any node without parents is called a *root node*, whilst those without children are *leaf nodes*, such that the root nodes represent the original causes and leaf nodes the final effects.

While qualitative relations among variables are captured by the BN topology, the conditional probability distributions of nodes *quantify* these relations. For discrete variables, the probabilistic uncertainties or inter-nodal dependencies are reflected in the corresponding *conditional probability tables* (CPT). Each node owns a single conditional probability table, which enumerates over all the possible combinations of that node's parents' values and records the conditional probability of that node taking on each of its specified values under each of those parental combinations. For a root node, its conditional probability table is composed of just one single row that contains all its known prior probabilities.

We can now visualize a simple Bayesian network in Figure 2.3, which we have adapted and modified from the lung cancer example that was presented earlier in Chapter 2 of [65]. This sample network essentially captures the underlying probabilistic interrelationships among the

five interrelated factors of *the level of pollution, whether the patient smokes, chances of that patient developing lung cancer, whether the x-ray report indicates presence of lung cancer* and lastly, *whether the patient exhibits dyspnoea (shortness of breath)*.

It is not difficult now for us to further appreciate the nature of Bayesian networks as graphical representations of joint probability distributions. Consider a Bayesian network with n nodes, denoted as X_1 to X_n . A particular value in the joint probability distribution is then compactly represented by $P(x_1, x_2, \dots, x_n)$, where $X_i = x_i$, $1 \leq i \leq n$. Applying the *chain rule* of probability theory, we factorize the joint probabilities so that

$$P(x_1, x_2, \dots, x_n) = P(x_1) * P(x_2|x_1) * \dots * P(x_n|x_1, x_2, \dots, x_{n-1}) = \prod_{i=1}^n P(x_i|x_1, x_2, \dots, x_{i-1}) \quad (\text{Eq. 2.1})$$

Since the BN structure implies that the value of any node is only conditional upon values of its parent nodes (referred to as the notion of *conditional independence* from all ancestors except immediate parents), the previous joint probability function reduces to

$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i|\text{Parents}(x_i)) \quad (\text{Eq. 2.2})$$

Under the reasonable assumption that the underlying problem structure is such that not every node is connected to every other node, the BN representation of joint probability distributions provides a much more compact representation than exhaustively describing the probability of every possible joint instantiation over all the available domain variables.

2.2.2 Reasoning within Bayesian Networks

The BN is not only useful for modelling the domain and its uncertainties - it allows us to reason about the domain given the available observed *evidence*, via a flow of information commonly known as *probability propagation, probabilistic inference, or belief updating*.

A BN provides us with the full representation of probability distributions over all the network variables, and this enables us to *condition* our reasoning based on any observable subset of the variables, both from parent to child and from child to parent. There are in all three basic types of reasoning, namely *diagnostic reasoning, predictive reasoning, and inter-causal reasoning*. These three types of reasoning are depicted in that same order in Figure 2.4, which is a modified version of the corresponding figure in Chapter 2 of [65].

CHAPTER 2. LITERATURE REVIEW

Figure 2.4: Types of Reasoning, As Illustrated Using the *Lung Cancer* Example.

In *diagnostic reasoning*, we reason from symptoms to causes. For example, a processed x-ray result would update a doctor's belief about the patient's chances of having cancer, on whether s/he smoked, and also on whether the patient was exposed to toxic pollution.

In *predictive reasoning*, we reason about new evidence on causes to infer new beliefs on effects. For example, the knowledge that a patient smoked regularly would increase his or her chances of having cancer, as well as increase the doctor's expectations of symptoms including shortness of breath (dyspnoea) and an affirmative x-ray result for the patient.

Thirdly, we can also reason about the mutual causes and common effects in what we call *inter-causal reasoning*. For example, both pollution and smoking causes cancer, but knowing that a cancer patient is a regular smoker reduces the probability that the patient has been exposed to pollution. In other words, this knowledge that the patient smokes *explains away* pollution as a cause of the patient's current observed state of illness. Last but not least, we should note here that since any observable node may become a piece of evidence, the above types of reasoning are commonly employed simultaneously, e.g. it is not uncommon for the evidence to come from a node that is both a parent and a child.

Next, we explain the belief updating mechanism in BNs by describing in details the probabilistic inference process for a simple but illustrative scenario of three network nodes connected

CHAPTER 2. LITERATURE REVIEW

in a chain, i.e., $X \rightarrow Y \rightarrow Z$ (please refer to Chapter 3 of [65] for more details).

Suppose we are presented with evidence that variable X has value x , or $X = x$. Since Y is a child of X , the posterior probability or belief for Y can be read off Y 's conditional probability table. For Z , we can employ the chain rule of probability theory as described previously, by relying upon the *conditional independence* that is implied in the network.

$$\text{Belief}(Z = z) = P(Z = z|X = x) = \sum_{Y=y} P(Z = z|Y = y)P(Y = y|X = x) \quad (\text{Eq. 2.3})$$

In another scenario, suppose we do not know the value of X , but we know that $Z = z$. Updating the beliefs for Y would involve applying the *Bayes' Theorem* as shown below.

$$\text{Belief}(Y = y) = P(Y = y|Z = z) = \frac{P(Z = z|Y = y)P(Y = y)}{P(Z = z)} = \alpha P(y)\lambda(y) \quad (\text{Eq. 2.4})$$

where $\alpha = \frac{1}{P(Z=z)}$ can be computed as a normalizing constant since the beliefs for all values of Y must sum to 1. The term $P(y)$ refers to the prior probability for $Y = y$, and $\lambda(y) = P(Z = z|Y = y)$ is the *likelihood* probability of $Z = z$ given that $Y = y$, which can be directly read off the conditional probability table of Z since Y is a parent of Z .

In the chained network, Z is conditionally independent of X given Y . By applying Bayes' Theorem and the chain rule of probabilities, we can compute X 's beliefs as follows

$$\begin{aligned} \text{Belief}(X = x) &= P(X = x|Z = z) = \frac{P(Z = z|X = x)P(X = x)}{P(Z = z)} \\ &= \frac{\sum_{Y=y} P(Z = z|Y = y, X = x)P(Y = y|X = x)P(X = x)}{P(Z = z)} \\ &= \frac{\sum_{Y=y} P(Z = z|Y = y)P(Y = y|X = x)P(X = x)}{P(Z = z)} = \alpha P(x)\lambda(x) \end{aligned} \quad (\text{Eq. 2.5})$$

where $P(x)$ is the prior value for $X = x$, and $\lambda(x) = \sum_{Y=y} P(Z = z|Y = y)P(Y = y|X = x)$ is the likelihood value which can be easily computed by summing the product of the corresponding probability values directly read off the conditional probability tables of Z and Y respectively, since Z is a child of Y and Y is a child of X . $\alpha = \frac{1}{P(Z=z)}$ is a normalizing constant since the beliefs of all values of X must indubitably add up to unity. According to [65], inference algorithms such as Kim and Pearl's *message passing algorithm* that are used for more complex structures than the chained nodes scenario generally involve the repeated application of the two fundamental components of the Bayes' Theorem and the chain rule of probability theory, as illustrated via the above simple chained BN example.

2.2.3 Bayesian Networks in Context-Aware Systems

As explained at the start of Section 2.2, a Bayesian network (BN) is a general graphical structure that allows us to represent and reason about an uncertain domain [65]. A BN can therefore be used to model the dependencies and uncertainty among contexts in a context-aware system (Section 2.1), in which case the nodes in the BN naturally represent the context parameters.

To model the inherent uncertainty in context inputs arising from imperfections in sensors and other sources of contexts, Gu et al. [66] had added dependency and probability markups as extensions to their context ontology defined in the W3C-endorsed Web Ontology Language (OWL). Their *manually-defined* context ontology could then be mapped directly into the topology or structure for a BN. The translated Bayesian network was then trained on data to learn the probabilistic distributions for the various nodes corresponding to the various context information, after which the resulting Bayesian network was used to reason about the contexts.

BN has many other meaningful applications in context-aware systems. For example, Lacey and MacNamara [67] developed a BN-based context-aware mobility aid for the elderly blind. Their robotic “smart walker” machine, PAM-AID, was designed to help frail and elderly blind users navigate safely when indoor. The onboard controller and the user shared control of the aid. PAM-AID assumed that users had a good topological idea of their environment, and supported them by announcing landmarks, warning of obstacles and performing local navigation. A Bayesian network was used to combine the user’s inputs with sensor-based feedbacks to estimate the user’s navigation goals, thus detecting any input errors. The resulting action selection mechanism had a very simple user interface, and this was critical for the elderly and blind.

In all these prior works, the causal dependencies were manually-crafted, i.e., none of these works had discovered the *user-specific* dependencies among contexts by learning from data, which we set out to do in this thesis. Like in the above works, the use of Bayesian network in our study is motivated by the efficient probabilistic reasoning capability of Bayesian network, as well as its graphical superiority in representing the causal dependencies among the context parameters. These are the key advantages which we believe would make it possible to construct Bayesian network-based recommender systems that are both robust to noisy or missing inputs, and also offer comprehensible probabilistic explanations for the resulting recommendations.

2.3 Recommender Systems

A *recommender system* [1, 68] is an application that is capable of ranking a list of similar items with consideration for one or more given criteria. In the general sense, classification systems, or classifiers such as those used for cancer detection and disease subtype classification, are simply particular instances of recommendation systems where inferences or prediction outcomes based on presented inputs represent the system's recommendations.

Recommender systems excel in environments where the amount of information is too overwhelming for any individual to survey manually, e.g. in information retrieval, or in tourist guides. As these systems are able to provide individualized recommendations to users faced with insurmountable number of options, recommenders in various forms have been the interest of many researchers, especially with the proliferation of mobile e-services (e.g. [4, 5]). The two main recommendation techniques in use today are the *collaborative filtering* approach [69] and the *content-based filtering* approach [70], although a number of other techniques exist, including many hybrid methods that have combined collaborative filtering with content-based and other techniques to improve recommendation [71–76].

2.3.1 Collaborative Filtering Systems

Collaborative filtering [69] works by seeking the opinions of a community of users to assist individuals in that community to better identify contents of interest from a potentially overwhelming array of choices. A collaborative filtering recommender compares its users for similarity in rating profiles, and returns items that similar users have rated highly. It relies on majority judgment within the community, and is largely based upon the notion of stereotypical behaviors and common interests. Like their content-based counterparts, typical collaborative filtering user profiles consist of vectors of items with ratings, which are updated as the users interacts with the recommender, although there are systems that compensate for the user's interest drift by discounting ratings over time, e.g. [77].

Examples of collaborative filtering recommender systems are many as this is the most mature and successful technique in use today, especially when some features of the items themselves are taken into account in addition to the similarity among users. These include the Grou-

pLens systems for collaborative filtering recommendation of Usenet news and movies [78, 79], the collaborative music recommender system by [74] that integrated the audio features of music into the collaborative filtering framework, and the item-to-item collaborative filtering recommendation system behind the successful Amazon.com [73].

2.3.2 Content-Based Filtering Systems

Stemming from early research in information filtering [80], *content-based filtering* recommender systems [70] suggest to users items that best fit their criteria. A content-based recommender learns from the properties of those items the user has rated highly in the past, and suggests items similar to those. The system understands the features describing the items of interest and is able to rank the items in terms of their relevance to the user's criteria. Such systems usually maintain user profiles that are highly specific to the problem domain, e.g. profiles in *research-paper* recommenders typically reflect only the user's research interests, although the sharing of profiles among autonomous agents had been suggested [81]. The user profiles in content-based recommenders are long-term models, and are updated as evidence about user preferences become available over time [72].

There are a number of existing content-based filtering recommender systems. They range from news recommendation systems like the newsgroup filtering application NewsWeeder [82] (which used the words in the news as features), book recommendation systems like LIBRA [83] (which recommended books to users based on the related information extracted from web pages at Amazon.com), to music recommendation systems such as the MusicSurfer system built by Cano et al. [84, 85] (which recommended songs to users based on their audio features).

2.3.3 Context-Awareness in Recommenders

A shortcoming of traditional recommendation techniques like content-based filtering and collaborative filtering is that they do not take into account situational information, which can seriously reduce their effectiveness. For instance, suppose that a user requests to “*recommend me a restaurant with Japanese food*”. A restaurant recommender selects a place with Japanese food, but do not realize that the user would have preferred a nearer place as it is raining. Clearly

CHAPTER 2. LITERATURE REVIEW

the system is not able to provide the best recommendations due to its detachment from the relevant situational information that characterizes the user-application interaction, or an *awareness of the context* [28] as explained in Section 2.1.

Over the last few years, many researchers had actively tried to realize the potentials of context-awareness, e.g. in the field of context-based information retrieval [86]. However, previous works on context-aware computing mostly supported only *static* context (e.g. [7, 13, 15, 55, 57–59]). In all these applications, the context set for consideration was *static*, i.e., the composition of context was fixed at design time and no attempt had been made to dynamically reduce or expand this set of contexts for a user via learning through usage.

Trevor et al. [14] presented an approach for automatically retrieving contacts based on the user's current context. The two main components were a capture component that captured contact information and stored it in a database, and a recommender component that analyzed the user's context to proactively recommend potentially relevant personal or shared contacts. The recommender front-end functioned as a toolbar in MS Outlook and IE, and monitored users' emails and web use to recommend them relevant contacts at appropriate times. It analyzed the transmitted text and used its matching algorithm to detect explicit occurrences of contact information fragments that matched entries in the system's contact database. To protect the user's privacy, interactions with the server were logged anonymously, and transmitted text was not persisted. Results were presented in a non-obstructive manner, and users could opt out of proactive recommendations easily.

In an earlier paper [16], we proposed a generic framework to enhance existing recommender systems with a task-oriented, context-aware model, and a learning approach that enabled the dynamic optimization of contexts for making recommendations to particular users. Applications benefit from an intelligently-selected set of contexts in a myriad of ways. For example, the application designer may have specified contexts not regarded as relevant by a user. By using the support vector machine (SVM) as a linear binary classification method for learning the relative weights of contexts, the proposed system pruned away the least relevant contexts while expanding it with more relevant ones overlooked by the designer. By instantiating only the most significant set of contexts, recommendation resources and time would be saved and the resulting advices would be more accurate.

2.3.4 Bayesian Networks for Recommendations

Use of Bayesian network (BN) is not new to recommender systems. Breese et al. [87] proposed a probabilistic collaborative filtering approach that learned BN from transactional data, with each node representing an item on sale, or a *commodity*. Likewise, Ji et al. [4, 88] proposed to learn a BN from a customer's shopping history, such that each BN node represented one commodity. The commodities were then recommended through probabilistic inference according to the last known shopping actions. All these works learned BN to capture the purchase patterns amongst the commodities. A drawback of this approach is that it suffers from the *sparsity* problem, where data points are often far fewer than commodities. More importantly, just the purchase patterns amongst specific commodities were learned from past transactions, without exploiting the potentially useful causal dependencies amongst other context parameters.

A massive number of papers had applied Bayesian network to classification problems, as many researchers recognized the unique potentials of BN in handling incomplete/noisy data and modelling variable interactions. Such properties make the BN an effective classifier; not only can the BN predict a target class based on partial subset of the attributes, it also yields insight to the attribute relations and their contributions to the inference. In particular, Bayesian networks had been used extensively in biological data mining, where open-problems abound. For instance, the more common tasks include biological function prediction and disease subtype classification, both of which usually require the analysis of large volumes of noisy observation data, such as gene and protein expressions from high-throughput experiments.

Needham et al. [89] applied Bayesian networks to two classification tasks in bioinformatics: 1) predicting whether a single nucleotide polymorphism (SNP) (a single mutation in the DNA chromosomes) effects, or brings about, protein function, and 2) predicting the interface between two interacting proteins (these provide important clues to functions of proteins). In the first problem, the Bayesian network structure was learned on data from one protein while the parameters of the model were learned from another. In the second problem, additional arcs or links were added to the learned network structure based on expert knowledge. The BN yielded an improvement over previous machine learning methods including decision trees and support vector machines on these problems.

The probabilistic integration of data from heterogeneous sources seemed to have taken the center stage in the last few years. Troyanskaya et al. [90] presented a MAGIC system that used a Bayesian network to combine data from diverse sources (including microarray methods) to predict if two proteins are functionally related, i.e., whether they are involved in a common biological process. In their system, both the structure and prior beliefs of the BN were elicited from domain experts in microarray analysis and molecular biology. Along the same line of approach, Huttenhower and Troyanskaya [91] made use of expert-elicited Bayesian networks to model and to quantify the functional relationships between proteins based on data from heterogeneous sources. Their predefined network structures were either parameterized with expert estimates or using probabilities learned from data.

Barutcuoglu et al. [92] developed a Bayesian framework that combined the outputs from multiple classifiers according to some predefined functional taxonomy constraints. A support vector machine (SVM) classifier was trained for each class, and the predictions from these SVMs were combined in a Bayesian network to improve the overall accuracy. Since their Bayesian network structure was directly mapped from the functional taxonomy, the resulting classifier was consistent with the functional hierarchy, e.g. it upholds the constraint that a gene cannot serve a child function without belonging to its parents.

2.4 Chapter Summary

Section 2.1 of this literature review deals with the topic of context-awareness in computing. The section begins with a concise definition of the underlying concepts, provides a brief discussion of outstanding issues in the area, and ends with a survey on the numerous existing context-aware applications. In Section 2.2, we introduce the Bayesian network (BN) to the reader, and also describe some of the previous applications of BN in context-aware systems. Section 2.3 discusses about the recommender systems in general, starting with brief introductions to the two main recommendation techniques. Subsequent parts of Section 2.3 examine the applications of context-awareness in recommenders, as well as the use of BN with respect to recommendation problems including bio-data classification.

Chapter 3

Learning the Minimal Set of Contexts and Exploiting the Context Dependencies for Robust Recommendations

An awareness of contexts enhances traditional recommendation techniques by taking into account the situation of interaction surrounding user ratings, e.g. [9] and [93]. In Section 2.2.3 and Section 2.3.4 of the previous chapter, we reviewed the prior applications of Bayesian networks (BN) in context-aware systems and recommendation systems, respectively. In all these works, either the causal dependencies were crafted manually and therefore not user-specific (e.g. [66, 67]), or contexts other than past purchases were not exploited [4, 87, 88]. Thus, they cannot identify the minimal set of contexts, nor can they readily overcome noisy context inputs.

We leverage upon context-aware content-based recommender system (Section 3.1.1) with Bayesian network learning (Section 3.1.2) to address these challenges, where the *user* asks the system for recommendations on available *items* based on the *user context* and *item attributes*.

Definition (User Context): “User context” includes higher-level contexts like a user’s food preference, and more primitive situational contexts like the weather. It characterizes the user’s behavior and situation, and is portable across applications.

Definition (Item Attributes): In a recommender system, “item attributes” are the attributes that directly describes the item for which the recommendation is wanted. The item attributes are specific to the recommender application being considered.

Definition (Minimal Context Set): A minimal context set contains the smallest number of context parameters that is sufficient for a specific user and application.

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

Using a small restaurant recommender with m users and n items (i.e. restaurants) as an example, suppose there are two user contexts in this system namely *desired_food* and *is_raining*, two item attributes namely *has_desired_food* and *is_nearby*, and two typical user concerns, *food* and *convenience*. Given past observations on these parameters and the previous restaurant ratings from a specific user, our challenges are to 1) identify the minimal set of important contexts for a specific user (e.g. a certain user u_a may be only concerned about the food, while a second user u_b is concerned about both food and convenience - only *desired_food* is important for u_a , whilst both contexts are important to u_b), and 2) to capture and exploit the causal dependencies among this minimal set of important contexts to effectively overcome noisy context inputs (e.g. the *is_raining* input may be wrong 20% of the time, making recommendation for u_b difficult).

The rest of this chapter is organized as follows. Sections 3.2 and 3.3 present our approach to the first and the second challenge, respectively. To learn the minimal context set, we iteratively trim the learned Bayesian network model until all remaining parameters are significant or has a direct connection to the prediction variable (in the example above, we learn that user u_a is only concerned about food, unlike user u_b); to enable the recommender system to handle erroneous context inputs, we discover which are the erroneous contexts, estimate their individual error rates, and automatically compensate for the uncertainty in their values during recommendation (for example, we discover that the user situation context input for *is_raining* is missing or is wrong 20% of the time, and we exploit the causal dependencies between context *is_raining* and user u_b 's desire for spicy food to overcome this). Sections 3.4, 3.5 and 3.6 validate the proposed approaches on a restaurant recommender application and a web page recommendation data set.

3.1 Preliminaries

3.1.1 Context-Aware Content-Based Recommender Systems

By eliciting recommendations from users with rating profiles similar to the target user, collaborative filtering sidesteps the need to understand exactly what are the factors that make a certain user prefer a particular item, at the expense of not being able to personalize these recommendations to unusual users. In contrast, content-based techniques use the attributes or characteristics

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

of the items to make recommendations, which enables them to recommend even unrated items to users with unique interests and to provide explanations for all their recommendations [83].

Recommendation by collaborative filtering is not scalable; in order to determine the most similar users, one has to examine all m users and up to n items for each user, where n is the number of available items (complexity $O(m + n)$) [73]. On the other hand, as the typical user vector only contains a small number of rated items based on which content-based techniques can search for similar items to recommend, content-based recommendation scales well (complexity just $O(n)$) [73]. Furthermore, collaborative filtering suffers from the cold start problem, i.e., the difficulty of recommending new items and recommending to new users with very few ratings [94]. In contrast, for its purpose of searching for similar items, content-based recommenders only require a handful of ratings to circumvent the cold start problem for new users [94], and it has no issue with recommending new items with known attributes. As such, we leverage the *context-aware content-based recommender* to address the challenges listed in Chapter 1. A formal definition of *context-aware content-based recommenders* is given below.

The goal of a context-aware content-based recommender is to predict the utility of an unrated item, for a user $u_a \in U$, based on her rated items I_{u_a} and the context in which those ratings, or *scores*, are recorded. The advices are typically presented in two forms:

- **Prediction:** a number $P_{a,j}$ that expresses the predicted utility of item $i_j \notin I_{u_a}$ for the user u_a . $P_{a,j}$ would have the same scale (e.g. from 1 to 10) as ratings from u_a .
- **Recommendation:** a top- N list of available items, unrated before by the user u_a , for which the recommendation model predicts u_a will have the greatest preference.

Figure 3.1 shows a schematic diagram of the context-aware content-based recommendation process. The ratings matrix A records past ratings for items by users. A record of the rating by a user u_a on an item i_j is a triplet $\langle \mathbf{c}_{a,j}, \mathbf{f}_{a,j}, r_{u_a,i_j} \rangle$. This triplet comprises the vector of user context $\mathbf{c}_{a,j}$ and the vector of item attributes $\mathbf{f}_{a,j}$ for that interaction, as well as the given rating value r_{u_a,i_j} . For example, in restaurant recommendation, item attributes or item contexts might include whether vegetarian food is available, as well as whether an eating place is crowded.

Figure 3.1: The Context-Aware Content-Based Recommendation Process.

User contexts might include whether the user is a vegetarian, and whether he or she minds a crowded restaurant at the time of interaction.

Whereas collaborative filtering considers all m rows of ratings to recommend for user u_a , here only items rated by u_a are considered. Each of these records provides an example for learning user u_a 's profile, comprising of the vector of user context inputs, the vector of item attribute inputs, and the recorded rating, or the *score* value (the class variable).

From the data, the context-aware content-based algorithm computes, for an unrated item i_j , $P_{a,j}(S_j = v \mid \mathbf{c}_{a,j}, \mathbf{f}_{a,j})$ for $v \in R^S$, where R^S are values that score S can take (e.g. v is one of values “1” to “5” for a 5-point rating scale). This is the posterior probability that user u_a gives a score of v to item i_j with context vector $\mathbf{c}_{a,j}$ and attribute vector $\mathbf{f}_{a,j}$. The algorithm computes this for each item, and recommends those with highest ratings.

Based on this formalism of context-aware content-based recommenders, the objectives of the techniques presented in this chapter are 1) to automatically discover, for user u_a , a *minimal set* that contains only those contexts important to u_a , 2) to harness the causal dependencies among the context parameters to effectively overcome noisy context inputs.

3.1.2 Learning Bayesian Networks from Data with CaMML

Causal dependencies among variables in a domain can be effectively modelled within the directed acyclic graphical model of a Bayesian network. These dependencies are captured qualitatively by the network structure, i.e., the arcs linking the network variables (*nodes*), and quantitatively by the table of conditional probabilities that is associated to each node.

Heckerman [95] stated that the local distribution functions in Bayesian network (BN) are in fact classification models, and past works [96, 97] had argued that, given complete prediction data, neither decision tree nor BN would significantly outperform each other. However, unlike decision tree, a BN encodes the causal dependencies among its parameters. The results we present in this chapter clearly demonstrate that this makes BN more suitable than J4.8 decision tree for compensating missing and erroneous context inputs.

Most prior works rely on Bayesian networks that are either manually-crafted or translated from existing ontologies [66]. However, as causal dependencies and relative importance of the parameters are user-specific, we need techniques to automatically learn the network structure, and to parameterize the learned network based on data. In this work, we adopt the *CaMML* program [98] for automatic Bayesian network learning from data.

CaMML [98] stochastically searches over the entire space of possible causal models to identify the best model h that maximizes a Minimum Message Length (MML) posterior metric [65]. The MML structural posterior metric that is used in the *CaMML* program is defined as $P_{MML}(h) = e^{-I_{MML}(h)}$, for $I_{MML}(h) = \log N! - \sum_{i=1}^m \log p_i - \sum_{j=1}^n \log(1 - p_j)$, where N is the number of nodes in the model h , $(m + n) = (N)(N - 1)$ is the total number of possible directed arcs in h , p_i represents the prior probability for the existence of a directed arc i in h , and $(1 - p_j)$ reflects the prior probability for a possible arc j to be absent from h [65].

CaMML performs a *Metropolis search* in the space of all possible models. For each real model it visits, it computes a representative model and counts only on these representatives to overlook the trivial variations. This grouping of real models is necessary in practice because the space of real models is exponential in the number of nodes, and each model would otherwise get only a very small number of visits even for the sizable data sets [65]. The MML posterior of each representative is computed as the total MML posterior of its members. This total posterior approximates the probability that the true model lies within the MML equivalence class of the representative model. The best model is therefore the representative model that has the highest MML posterior. Since the number of possible models is factorial in the number of nodes N , the theoretical worst-case complexity of the *CaMML* learning process is $O(N!)$, although in practice the average complexity using the stochastic Metropolis sampling should at most be exponential in the number of nodes. For more details, please refer to [98] and [65].

Figure 3.2: A Two-Tiered Context Model.

3.2 Learning the Minimal Set of Contexts

3.2.1 A Context Model for Effective Learning

As discussed earlier in Section 3.1.2, the time complexity of the Bayesian network learning algorithm is dependent on the number of network parameters. This is a concern as there can often be far too many context parameters eligible for consideration in a recommender. Also, many contexts are acquired from sources that can become intermittently unavailable and require alternatives to be used, e.g. in service-oriented frameworks (e.g. [31, 99]). To achieve a compact and more manageable set of learning parameters, we propose the use of *domain-specific markers* popular amongst clinical (especially cancer) research works.

Our two-tiered context model based on the concept of markers is presented in Figure 3.2. In this model, a lower tier handles the *user context* and the *item attributes*, where the *user* refers to the person who is asking for recommendation, and the *item* refers to one of the many choices for which the recommendation is wanted. We define *user context* and *item attributes* as follow.

Definition (User Context): “User context” includes higher-level contexts like a user’s food preference, and more primitive situational contexts like the weather. It characterizes the user’s behavior and situation, and is portable across applications.

Definition (Item Attributes): In a recommender system, “item attributes” are the attributes that directly describes the item for which the recommendation is wanted. The item attributes are specific to the recommender application being considered.

With reference to Figure 3.2, an upper tier handles the *item markers* and the *item score*. In a recommender system, *markers* are the common concerns of typical users of that system, which makes them suitable for evaluating the items. We define *item markers* and *item score* below.

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT
DEPENDENCIES FOR ROBUST RECOMMENDATIONS

Definition (Item Markers): An “item marker” is a Boolean function that takes in several specific user contexts and item attributes as parameters, to determine whether a certain condition of concern is satisfied by each item. As per the item attributes, markers are specific to the recommender application being considered.

Definition (Item Score): The “item score” for a particular item refers to the rating that is provided by the user to reflect her assessment on the suitability of that item.

To give a better understanding of markers, let us look at some example recommendation problems. For instance, being a multi-racial and culturally diverse city, Singaporeans are often concerned about whether a restaurant offers certain types of food, such as Western, Muslim, Indian and Chinese. People who drives would be concerned as to whether it is convenient to drive or park at a restaurant, and many people would have their own preferences for restaurants which support cashless payment, which allow or disallow smoking, which are open-air or air-conditioned, and which are relatively well-maintained or clean. These concerns naturally form the markers that inject realism into our restaurant recommender application of Section 3.4.

To give more examples, let us consider a recommender designed to promote teenage readership of books from a public library. Some typical concerns of teenage readers would include the genre of the book, how popular it is with other teens, et cetera. All these factors can thus be incorporated as markers in the book system, i.e., whether a certain book matches a patron’s favorite genre (e.g. sci-fic, romance or fantasy), or how popular a book is amongst peers, etc. The bottom line here is that a responsible system designer is expected to understand the need of her typical target users - it would then not be unreasonable or difficult for her to identify the most salient factors, and thereby create or annotate useful markers based on all such concerns.

Since the defined markers are typically far fewer than the relevant context parameters, learning on the markers allows us to discover the minimal marker set from fewer examples. We can then recognize those context parameters that feed into this minimal marker set as being important to the user. In principle, the Boolean functions for computing the marker values can be generated automatically. For instance, a similar user-adaptation problem had been previously addressed by bootstrapping the recommender with user profiles sampled from a probabilistic

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT
DEPENDENCIES FOR ROBUST RECOMMENDATIONS

model built based on prior knowledge [100]. In those applications without defined markers, the two-tiered model applies directly on the context parameters. In such cases, we learn the minimal set of parameters and include parameters connected to the minimal set as the second tier. Doing so effectively captures their causal dependencies, as it would be demonstrated via the web page recommendation problem that is presented in Section 3.6. For the rest of this section, we describe our techniques assuming that markers are available for the recommender system, but the proposed techniques are equally useful in applications with no defined markers.

3.2.2 Identifying and Learning on the Optimal Set of Predictors

We have developed an iterative procedure for learning a Bayesian network from data with only the minimal set of context parameters. First, we learn with just the markers and the score variable to find the minimal set of important markers. Learning on all the markers initially, we retain those markers that are connected to the rating, or the *score* variable, for a second learning. In this manner, the procedure iteratively trims away markers until the remaining markers are all connected to the score variable. We can now include those context parameters and item attributes that feed into the minimal marker set as part of our learning parameters. The Bayesian network learned using this procedure effectively captures the causal dependencies among the context parameters important to that user. The above minimal set learning procedure has been summarized in Procedure 1, where we assume the two-tiered context model shown in Figure 3.2. A variation of this minimal set learning procedure for domains without defined markers is presented in Procedure 2.

Procedure 1 Minimal Set Learning With Domain Markers

Input: Data set (D) with marker set (M), context set (C)
and the score variable (s) as features.

Output: Learned Bayesian network (BN) associating the
minimal set of markers and contexts to s .

repeat

Learn BN on D using only features M and s .

$\forall n$, node n not directly connected to s in BN, $M = M - \{n\}$.

until all markers are directly connected to s in BN.

Learn BN on M , s , and those $c \in C$ that feed into M .

return BN.

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

Procedure 2 Minimal Set Learning Without Domain Markers

Input: Data set (D) with context set (C)
and the score variable (s) as features.

Output: Learned Bayesian network (BN) associating the
minimal set of contexts to s .

repeat

Learn BN on D using features C and s .

$\forall n$, node n not directly connected to s in BN, $C = C - \{n\}$.

until all contexts are directly connected to s in BN.

Learn BN on C , s , and those nodes directly connected to $c \in C$.

return BN.

Our proposed context model eliminates the need to learn on the large set of context parameters in applications where a far smaller set of markers are available. Also, unlike in those prior works that are described in Section 2.3.4, which model specific commodities as network nodes, we learn a compact and explanatory Bayesian network to capture the user-specific causal dependencies among the context parameters. As such, we can predict ratings even for new items, without having to add new nodes or to relearn the network.

3.2.3 Performing Predictions using Learned Bayesian Network

Using the *Netica-J* API [101], we predict the *score* by feeding the values of the user context and item attributes to the learned BN, and allowing the beliefs of the marker values and the item score to be updated. We then recommend the highest-scoring items to the user. Once *CaMML* has recorded the learned model in the Netica format, invoking *Netica-J*'s belief updating mechanism for a prediction involves the following straightforward steps:

Step 1: For an unrated item i_j , present its current context vector $\mathbf{c}_{a,j}$ and attribute vector $\mathbf{f}_{a,j}$ for a user u_a to the learned Bayesian network.

Step 2: Let the Bayesian network update all $P_{a,j}(S_j = v \mid \mathbf{c}_{a,j}, \mathbf{f}_{a,j})$, where v represents each of the values that item score variable can take on.

Step 3: Predict, for the unrated item i_j , the score value v having the highest posterior probability value as reflected by $P_{a,j}(S_j = v \mid \mathbf{c}_{a,j}, \mathbf{f}_{a,j})$.

Netica uses the fastest known algorithm for exact general probabilistic inference in a compiled Bayesian network, which is known as *message passing in a join tree* (“*junction tree*”) of *cliques*. For details of this algorithm, please refer to the software’s manual [101] and the more technical discussions in [102]. As belief updating is done once for each item in each round of recommendation, time complexity of each recommendation is $O(n)$, where n is the number of items. Empirically, the BN inference mechanism is very efficient. In the experiments, hundreds of predictions are completed within just seconds on a PC.

3.3 Handling Missing and Erroneous Context Inputs for Robust Prediction

3.3.1 Handling Missing Context Inputs

In most supervised learning techniques, including decision tree and the support vector machine, missing values in the test examples are typically handled by filling in the blanks. We can build a prediction model for each missing attribute to estimate its missing values. Otherwise, we can fill in missing values with aggregate values (e.g. mean, mode) that are determined from the training set. Besides these, there are other sophisticated techniques for handling missing values in the statistical literature. One popular approach is *Multiple Imputation* [103], which involves the processes of imputation, analysis and pooling. Good references on this and other statistical methods to handle missing data include [104, 105].

Whereas all the above techniques require the manipulation of data to prepare them for prediction, in a Bayesian network, partial evidence can be entered during prediction, and BN would compensate for the unspecified inputs using its captured causal dependencies.

3.3.2 Handling a Single Error Context

Our procedure for handling erroneous context inputs is shown in Figure 3.3. The training set comprises of the parameters that are identified through our iterative learning procedure of Section 3.2.2. The error-handling procedure learns an initial Bayesian network from this training set with context in error. Predicting on the training set with this network, the procedure iterates

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

Figure 3.3: Procedure for Handling Erroneous Context Inputs.

over the individual context to identify the erroneous context, such that the percentage of misclassified training examples for each context is its estimated error rate. The procedure retains the correctly classified examples, or the *hits*, and learns a more accurate Bayesian network on these examples. This final Bayesian network then takes into account the estimated error rate when predicting on unseen examples.

The sub-procedure for automatically identifying a single erroneous context and cleansing the training set is summarized into Procedure 3. First, a process identifies the context that is in error and estimates its error rate; secondly, the training set is “cleansed” by discarding those training examples that are misclassified for that context. The time added by the Steps 1 and 2 is linear in the size of the training set, K , due to the repeated predictions on the training examples. Letting the number of context parameters be p , the complexity of these steps is $O(pK)$. This is efficient in practice as K is usually small (the typical user would seldom have seen too many examples). With reference to Section 3.1.2, *CaMML* learning has a worst-case time complexity of $O(N!)$, where N is the number of nodes considered in learning the BN. The overall complexity of Procedure 3 is thus $O(pK + N!)$, with the bulk of the time taken up by the

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

CaMML learning from data.

Procedure 3 Handling a Single Unidentified Error Context with an Unknown Error Rate

Input: Initial Bayesian network ($BnIn$) that is learned on the original training set with errors ($dataIn$).

Output: Final BN ($BnOut$) learned on cleansed data ($dataOut$), the error context ($errCtx$) and its est. error rate ($errEst$).

Step 1: Identify $errCtx$ and find $errEst$

Set the first context, C_1 , as $errCtx$.

Treat C_1 as missing and predict its values in all of $dataIn$ using $BnIn$.

Set $errEst$ to the proportion of $dataIn$ for which C_1 is misclassified ($P_{err}(C_1)$).

for each remaining context C_i **do**

 Treat C_i as missing and predict its values in all of $dataIn$ using $BnIn$.

if $P_{err}(C_i) > errEst$ **then**

 Set $errCtx$ to C_i .

 Set $errEst$ to $P_{err}(C_i)$.

Step 2: Cleanse $dataIn$ of suspicious examples

for each example in $dataIn$ **do**

 Treat $errCtx$ as missing and predict its value using $BnIn$.

if the predicted value of $errCtx$ differs from its labelled value **then**

 Remove this example from $dataIn$.

{If resources permit, **Step 2** simply recalls and removes examples misclassified earlier in **Step 1**.}

Step 3: Complete the error context identification and data cleansing procedure

Learn $BnOut$ on cleansed training set $dataOut$ from **Step 2**.

return $BnOut$, $dataOut$, $errCtx$, and $errEst$.

Training Bayesian network on “cleansed” data makes sense as we expect better models when the training set contains fewer error examples. This intuition is an important motivation behind past outlier detection methods [106], as well as works that have extended the notion of “noise” to include examples that are irrelevant or weakly-relevant [107].

3.3.2.1 Identifying Error Context and Estimating Error Rate

In the data mining field, dealing with noisy or erroneous data usually involves identifying training examples with inconsistent values as *outliers* and removing these from the training set so that we learn an improved model [106]. In general, the outlier detection methods (e.g. distance-based, density-based, clustering-based) assign an outlier score to each example and remove those examples above some threshold [107]. For instance, [108] employed majority-voting to discard those training examples deemed to be mislabelled.

Drawing inspiration from the outlier detection methods, we learn a prediction model on the training set to predict on each context in turn, in order to identify those outlier examples where

the predictions differ from their observed values. For a start, we assume that only one of the contexts important to the user is in error, but we do not know which one it is, nor what is its rate of error. All we are initially presented with is the training examples that are noisy. Hence, we begin by using *CaMML* to learn an initial BN, which we then use to predict on each context in turn and estimate their individual error rates.

3.3.2.2 Cleansing the Training Set

Having estimated the individual error rates, we identify the context with the largest error as the one that is erroneous. We now have its training error rate (the proportion of training examples for which the initial learned BN prediction model has misclassified the erroneous context) as our “knowledge of context with largest error” (Figure 3.3). Each correctly classified training example is termed as a *hit*, and we separate these *hits* from the *misses* (those incorrectly classified training example) to get a reduced set of training examples. We cannot ascertain in practice if this reduced set of training examples is in fact error-free. What we hope to do is to remove as many of those examples with possibly erroneous context inputs, so that the final Bayesian network learned from the “cleansed” training set can better reflect the actual causal dependencies among context parameters.

3.3.2.3 Predicting on Score using Likelihood Findings to Account for Uncertainty in Context

After obtaining the cleansed set of training examples (the *hits* from the initial Bayesian network, i.e., those correct predictions on the identified error context), we learn a final BN for predicting the user ratings, or the *scores*, of unrated items. We can validate this final BN’s accuracy by predicting the user ratings for an unseen set of test examples. This test set should be noisy too, with similar noise characteristics as the original training set.

When predicting the user rating, the Bayesian network is able to take into account the estimated error rate, or the *uncertainty*, in the erroneous context. This is an important capability not available in other classifiers such as decision trees. In the normal inference scenario, where we are not aware of possible errors in the context inputs, we enter each input from the test example as a *specific evidence*. For example, suppose that the input values for the context

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

parameter “weather” can be “hot”, “rainy” or “fine”. Now, if we enter an input of “fine” as a *specific evidence*, a likelihood probability of 1.0 would be used for “fine”, whilst 0.0 would be used for the other two values. However, when there are errors in the inputs of a context, entering these erroneous inputs as specific evidence is likely to result in wrong inferences. Fortunately, Bayesian network allows us to specify such potentially erroneous evidence as *likelihood findings* instead of specific evidence.

Let the observation on the value of “weather” context be denoted as O . The *likelihood finding* for O is then given by $\{\text{prob}(O | \{\text{weather}=\text{“hot”}\}), \text{prob}(O | \{\text{weather}=\text{“rainy”}\}), \text{prob}(O | \{\text{weather}=\text{“fine”}\})\}$. For the example where we observe “weather” as “fine”, suppose we are also aware that this observation carries an estimated error probability of e . This means that each of the input values for “weather” might be wrong $e * 100\%$ of the time. This error probability for the erroneous context could be estimated using the training error rate discovered by our error-handling procedure. The likelihood finding for $O:\{\text{weather}=\text{“fine”}\}$ would then be given by $\{e * 0.5, e * 0.5, 1.0 - e\}$. This is because O may be wrong with a probability of e , and “weather” has only three states, so the probability of inputting the “weather” value as “fine” when it is actually “hot” or “rainy” is $e * 0.5$. Depicting S as the number of states of the erroneous context, we generalize this formulation of the likelihood finding from the estimated error probability e as follows:

Likelihood for the observed state = $1.0 - e$, and

Likelihood for any other state = $e * [1.0 / (S - 1)]$.

As the effect of entering an error rate of zero for each context is equivalent to entering a specific evidence (with a likelihood of 1.0 for the observed value and 0.0 for all others), this likelihood formulation is appropriate even when we are certain about the evidence. As explained in [65], the prior probabilities within the training set should be considered when computing likelihoods. However, as our restaurant recommendation data set covers all states of each context without bias, the input priors are roughly uniform. Results in Section 3.5.3 show that, by accounting for the uncertainty in the erroneous context, Bayesian network overcomes error context inputs to predict robustly on unseen examples.

3.3.3 Handling Multiple Erroneous Contexts

So far, we have assumed the knowledge that a single context is in error. In practice, we do not know how many contexts might be erroneous, so there is a need to make sure that our procedure is not restricted by this assumption. The proposed procedure for handling multiple erroneous contexts factors out the negative effects of the highest-error context one at a time until the estimated error in the remaining contexts is below some defined threshold. This error handling procedure has been summarized into Procedure 4. Since Steps 1 and 2 iterate over the set of p context parameters, the worst-case time complexity of this procedure is $O(p^2 K + (p+1)N!)$, where K is the number of training examples, and N is the number of nodes in the network. Most of this complexity ($O((p+1)N!)$) is due to the *CaMML* learning, which is highly sensitive to the size of the network as represented by N . As such, our iterative learning procedure (Section 3.2.2), which finds the minimal set of important parameters, can significantly improve the efficiency of error-handling.

Procedure 4 Handling Multiple Unidentified Error Contexts with Unknown Error Rates

Input: Initial BN ($BnIn$) learned on the original training set ($dataIn$),
and the minimum threshold for an error context ($errMin$).

Output: Final BN ($BnOut$) learned on cleansed data ($dataOut$),
the error contexts ($errCtxs$) and their est. error rates ($errEsts$).

Step 1: Identify $errCtxs$ and find $errEsts$

Identify the top most-erroneous context C_{top} as per Procedure 3.

Add C_{top} to $errCtxs$ and set $errEsts$ to error rate of C_{top} .

while min. $errEsts \geq errMin$ **do**

 Estimate likelihoods using $errEsts$.

 Entering the values of $errCtxs$ as likelihoods, identify the next most-erroneous context C_{next} .

 Add C_{next} to $errCtxs$ and set $errEsts$ to current error rates of $errCtxs$.

Step 2: Cleanse $dataIn$ of suspicious examples

while mean error on $errCtxs$ is significant under a statistical t-test **do**

 Remove the training examples that misclassify the context in $errCtxs$ with the largest error.

 Learn on this partially cleansed data for the next iteration.

Step 3: Complete the error contexts identification and data cleansing procedure

Learn $BnOut$ on cleansed training set $dataOut$ from Step 2.

return $BnOut$, $dataOut$, $errCtxs$, and $errEsts$.

3.3.3.1 Identifying Error Contexts and Estimating Error Rates

Our error-handling procedure described in Section 3.3.2 is extended as follows. First, we iteratively predict on the important contexts with all inputs entered as *specific evidence*, and we take note of the error rate for the single *most-erroneous* context. We again iterate over all the contexts, but now we specify all inputs for that *most-erroneous* context as *likelihood findings* computed based on its estimated error rate. The next most erroneous context is identified, and its error rate as well as that of the most-erroneous context are noted. We then iterate over all contexts using the likelihood findings for these two most erroneous parameters. This process is repeated until the estimated error rate of the next most erroneous context falls below a specified minimum error threshold. In this way, the procedure identifies the error contexts in sequence, using their estimated error rates as a reasonable basis for identifying the order in which these contexts are likely to be in error.

This iterative error discovery process is analogous to *Principal Components Analysis*. We first identify the most erroneous context, remove its dismal effects from the training set by inputting it with uncertainty in the next round, then look for the next most erroneous context and so on. Analogous to Henry Kaiser’s recommendation for retaining only principal components with eigenvalue exceeding 1.0, our procedure stops looking for error contexts when error estimate in the next most erroneous context is below some threshold. As users generally exhibit some reasonably small degree of natural randomness in behaviors, the minimum error threshold can help to filter out such background noises.

3.3.3.2 Cleansing Training Set by Removing Examples with Errors in One or More Contexts

We have to decide upon the amount of “data cleansing” before commencing on the final learning. Under the previous assumption of a single erroneous context, we simply remove those examples with wrong predictions on that context from among the training set. With multiple contexts in error, we have the alternatives of either removing misses for all the error contexts identified, or to remove just misses on the most erroneous context, learn a BN on the hits, and evaluate the model on an unseen validation set to decide whether to proceed with removing

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

misses on the next most erroneous context and so on. Removing all the training misses for all the error contexts presents the danger of *over-fitting* as the size of our training set dwindles, so the second approach provides us with a means to cross validate on the given examples in order to avoid over-fitting the training data.

We implemented the above second approach with certain modifications, so as to overcome the two limitations of having to set aside a validation set, and having to predefine a fixed threshold for deciding when to stop cleansing. In the first iteration of data cleansing, we perform a statistical t-test [109] to check if the mean estimated context error is statistically significant at some desired level (e.g. 0.01, 0.05). If so, we proceed to remove those training examples that have misclassified the single context with the largest error, learn on this partially cleansed training set for the next iteration, and so on. The data cleansing stops when the mean estimated error is not significantly different from 0.0 in any iteration. The experimental results for this procedure are presented in Section 3.5.5.

3.4 A Restaurant Recommender Application

We demonstrate how our proposed approach works in a restaurant recommender application that is named “*RR-II*” (our second restaurant recommender). In this application, we have created a sizable set of realistic user models that represent multiple users with differing considerations and context models when it comes to meals. The task of “*RR-II*” is to learn automatically from data the underlying causal dependencies among parameters as a Bayesian network for a specific user, and predict the scores for all candidate restaurants based on this learned network. To make this application realistic, a rich set of real-world user context parameters, restaurant attributes and typical user considerations or markers has been defined (Appendix B). Each user has her own preference rule for markers, i.e., a set of concerns important to her (Table 3.1), and a causal model which relates the corresponding user contexts, i.e., the user’s behavior in terms of the causal dependencies between her personal preferences and her situation (Appendix B).

For each user, we generate an observation or data example to simulate an instance of interaction between that user and the “*RR-II*” system, where the user rates each restaurant based on the parameter values present at that time. Each example comprises a set of context values

Figure 3.4: Context Model of Our Restaurant Recommender Application, *RR_II*.

adhering to the user’s representative causal model, a set of restaurant attribute values for each restaurant, the marker values derived using application heuristics from the context and restaurant attribute values, and the score assigned by that user to each restaurant (computed from the marker values according to the preference rule representing the user). During the evaluations, only these data examples are presented to the system. This enables us to validate whether our proposed techniques are indeed useful for reliably discovering from data the unique personal characteristics of different users and exploiting these to maintain robust recommendations.

Figure 3.4 shows *RR_II*’s context model, which implements the model of Figure 3.2. We have defined twenty-six relevant user context parameters, thirty restaurant attributes, and twenty-one markers (Appendix B). In a typical recommender, the number of contexts is much greater than the number of markers, so the advantage of not having to learn on all the contexts is even more significant in practice. The user contexts include the relevant aspects of a user’s current *situation* (e.g. the “*weather*”), as well as the relevant aspects of a user’s *preferences* (for example, the “*cleanliness*”). Restaurant attributes include “*category*” and “*is clean*”. As described in Section 3.2.1, the markers in *RR_II* are defined to represent typical user concerns like whether the user attire is appropriate for the restaurant, and are derived from the interactions between user contexts and item attributes. Each marker is a boolean variable, e.g. a restaurant is either “open” or “not open” at the time of user visit, and the restaurant ventilation either “matches” or “does not match” the user’s preference. Fifteen restaurants are available for recommendations.

The causal model for each user (Appendix B) is defined based on user contexts and not system markers to accurately reflect how the typical and logical users would consistently behave. For example, our first model in Table 3.1 represents a user who takes care to dress for different occasions. A set of causal rules models how such a user dresses up for different eating places. For example, s/he dresses consciously when visiting a club; otherwise s/he prefers to

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT
DEPENDENCIES FOR ROBUST RECOMMENDATIONS

stay casual when eating in a cafe or canteen. In addition, this user prefers to eat in open-air canteens for informal occasions because fresh air improves her appetite. The causal model that represents this user, User 1, is shown below. The user context values in each observation are randomly picked from their range of possible values while adhering to the user's causal model. For User 1, the values of all user contexts are randomly assigned, except for user attire situation and user ventilation preference, which are assigned values according to the preferred category.

Model 1 *Causal Dependencies Among Contexts for User 1*

```

if UP.category == "cafe" then
  US.attire = "casual"; UP.ventilation = "aircon";
else if UP.category == "club" then
  US.attire = "formal"; UP.ventilation = "aircon";
else if UP.category == "canteen" then
  US.attire = "casual"; UP.ventilation = "nonAircon";
else if UP.category == "dunCare" then
  US.attire = "casual"; UP.ventilation = "dunCare";

```

In each recommendation cycle, the user context values and restaurant attribute values are compared via heuristics to determine whether a marker should take on a value of 1 ("satisfied") or 0 ("not satisfied"). Examples of heuristics for three of the defined markers (M3, M11, M12) are given below. The prefixes "US." and "UP." indicate a *user situation* and a *user preference* context respectively, whilst "RA." refers to a *restaurant attribute*.

Heuristic M3 *userAttireIsAppropriate (M3)*

```

if RA.attireRequirement == "formal" then
  if US.attire == "formal" then
    valueof(M3) = "yes";
  else if US.attire == "casual" then
    valueof(M3) = "no";
else if RA.attireRequirement == "none" then
  valueof(M3) = "yes";

```

Heuristic M11 *matchesIsAirConditionedPref (M11)*

```

if UP.ventilation == "aircon" then
  if RA.isAirConditioned == "yes" then
    valueof(M11) = "yes";
  else valueof(M11) = "no";
else if UP.ventilation == "nonAircon" then

```

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

Figure 3.5: Causal Dependencies for Markers M3, M11 and M12.

```

if RA.isAirConditioned == "yes" then
    valueof(M11) = "no";
else valueof(M11) = "yes";
else if UP.ventilation == "dunCare" then
    valueof(M11) = "yes";

```

Heuristic M12 *isOfDesiredCategory (M12)*

```

if UP.category == RA.category then
    valueof(M12) = "yes";
else valueof(M12) = "no";

```

Figure 3.5 shows the defined causal dependencies corresponding to each of the markers M3, M11 and M12. Based on the corresponding derived marker values, we simulate that a user rates the restaurant, or assigns a *score* value to the restaurant, using the formula below:

$$score = \frac{\text{No. of Important Markers with value "1"}}{\text{No. of Important Markers}} .$$

This formula reflects the behavior of a logical user who rates each of the presented restaurants according to their overall performance on the minimal set of markers that is personally important to him or her. The formula for equal weighing of important markers that is used in this work is an example of the many possible scoring functions for a user, which of course can be much more complex and flexible for each idiosyncratic user. In fact, in typical application scenario, there would not be any such explicit scoring function - the users of the recommender system would instead have indicated their preference for each item using certain rating scales, after taking into account the values of markers important to them. Each example in our experiments thus comprises the user context and attribute values, the marker values derived using the heuristics as described above, and the user scores for the restaurants. During our evaluations, just as in the actual application scenario, only the data examples are presented for analysis, and the system has to learn to predict the score for each user solely based on these records.

3.5 Experimental Validation

3.5.1 Learning the Minimal Set of Important Parameters

The first set of experiments aims to verify that the minimal set of markers truly important to each user can be effectively recovered through Bayesian network learning from observations. In this and all the subsequent experiments on the *RR-II* restaurant recommender application, observations or data examples are generated based on a set of *preference rules* which represents different user logics in considering markers when scoring restaurants, and a set of *causal models* that defines the causal dependencies among the corresponding important contexts. The preference rules and their respective causal models are listed in Table 3.1, with one preference rule and one causal model for each user. Each preference rule is generated by selecting randomly from the set of available markers, and the causal model is generated accordingly by picking from a set of manually defined context dependencies those context dependencies that correspond to the markers in the preference rule. For each user, we generate examples spanning across all the possible values of the context parameters while adhering to her specific causal model (see Appendix B, Causal Models 1-20). For each example, the values of markers that are important to the user (according to her preference rule) are derived from the context values and each restaurant's attribute values, and used to score the restaurants as described in Section 3.4. For instance, according to the preference rule for *User 1* of Table 3.1, the values of markers “*userAttireIsAppropriate*”, “*matchesIsAirConditionedPref*” and “*isOfDesiredCategory*”, labelled as M3, M11 and M12 respectively, are used by *User 1* to score each restaurant.

For each rule, following the steps in our minimal set learning Procedure 1 of Section 3.2, we train a Bayesian network using a thousand observations generated as above, and take note of those markers directly connected to the *score* node within the learned network. We then train again on the same set of examples with all other markers removed. This process of automatic learning and pruning is repeated till a learned model has all its markers directly linked to *score*.

From our first round of learning on the training set corresponding to User 1, we obtain the Bayesian network that is shown in Figure 3.6.a. On the right-hand side of this figure, we see that only the four markers of M3, M11, M12 and M18 are connected to the score node.

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

Table 3.1: User Preference Rules and Models.

User	Preference Rule	Causal Model
1	M3, M11, M12	Model 1
2	M5, M10, M12, M15	Model 2
3	M10, M12, M14	Model 3
4	M5, M10, M12, M21	Model 4
5	M3, M5, M12	Model 5
6	M11, M5, M14, M17	Model 6
7	M12, M10, M19	Model 7
8	M17, M21, M13, M3	Model 8
9	M10, M18, M13	Model 9
10	M12, M5, M10, M13	Model 10
11	M15, M18, M12	Model 11
12	M15, M6, M17, M14	Model 12
13	M3, M12, M18	Model 13
14	M12, M21, M19, M3	Model 14
15	M13, M11, M16	Model 15
16	M3, M12, M6, M16	Model 16
17	M6, M19, M12	Model 17
18	M6, M10, M12, M17	Model 18
19	M13, M6, M12	Model 19
20	M18, M14, M12, M5	Model 20

3.6.a: In Round 1.

3.6.b: In Round 2.

3.6.c: In Round 3.

Figure 3.6: Discovered Bayesian Network in Each Round of Learning.

To verify if all these four markers are important for scoring, we retain just these four markers and the score variable to perform our second round of learning. From the resulting network as shown in Figure 3.6.b, only M3, M11 and M12 (the three nodes on the left) are directly connected to the score variable. We therefore retain only the score and these three nodes for our third round of confirmation learning. The resulting network of Figure 3.6.c shows that *all* the three markers that are supposed to be important to the first user (as stated in Table 3.1) are still directly connected to the score variable.

To evaluate the system's performance in discovering the correct minimal set of markers, we adapt the *F-measure* in information retrieval field with the following definitions:

$$Recall = \frac{No. of Important Markers Connected to Score}{Number of Important Markers}$$

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

$$Precision = \frac{\text{No. of Important Markers Connected to Score}}{\text{No. of Markers Connected to Score}}$$

$$F\text{-measure} = \frac{2(\text{Recall})(\text{Precision})}{(\text{Recall})+(\text{Precision})}$$

These modified definitions accurately reflect our context of analysis where a positive hit corresponds to an important marker being directly connected to the score node in the learned Bayesian network model. The *recall* value represents the proportion of truly important markers that are identified as important, while *precision* measures the proportion of those markers marked as important which are truly important. The *F-measure* is then the weighted average, or the harmonic mean, of *recall* and *precision*. So together, they measure how well the set of important markers matches those connected to *score*.

The results of learning from the data sets are given in Table 3.2(a) and Table 3.2(b). A further breakdown on the number of users who achieve a 100% measure in a particular round of learning is given in the last column of Table 3.2(c). For example, Table 3.2(a) and Table 3.2(b) show that the learned networks yield a F-measure value of 100% by the second round for Users 1, 2, 3, 5 and 13, while a 100% F-measure is achieved after just one round for User 4. This is reflected in Table 3.2(c) as one minimal set found in Round 1 and five in the second round. In addition, Table 3.2(c) shows the average recall, precision and F-measure across those users who have undergone each round of learning. For instance, only User 12 has required a fifth round of learning, so the recall, precision and F-measure values for Round 5 in Table 3.2(c) simply reflect her corresponding values in Table 3.2(b). Based on these results, our proposed minimal set learning approach reliably identifies the minimal set of parameters for all the twenty users.

3.5.2 Prediction with Missing Context Values

In our second set of experiments, we employ the minimal sets of important markers identified in our earlier experiments to verify that the prediction accuracy on *score* remains high despite missing context values. For each user, we prepare the same earlier set of a thousand observations to perform five rounds of two-fold cross-validation. In the first fold of each round, five-hundred examples are randomly chosen for learning the dependencies among contexts, and the remaining examples are used to test the prediction accuracy on score. In the second fold, we now train on the held-out set and use the previous training set for testing. In each

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

Table 3.2: Minimal Set Learning Performance.

(a) User 1 to User 11.					(b) User 12 to User 20.				
User	Round	Recall	Precision	F-measure	User	Round	Recall	Precision	F-measure
1	1	1.00	0.75	0.86	12	1	1.00	0.19	0.32
	2	1.00	1.00	1.00		2	1.00	0.50	0.67
2	1	1.00	0.67	0.80		3	1.00	0.67	0.80
	2	1.00	1.00	1.00		4	1.00	0.80	0.89
3	1	1.00	0.50	0.67		5	1.00	1.00	1.00
	2	1.00	1.00	1.00	13	1	1.00	0.14	0.25
4	1	1.00	1.00	1.00		2	1.00	1.00	1.00
	2	1.00	1.00	1.00	14	1	1.00	0.19	0.32
5	1	1.00	0.60	0.75		2	1.00	0.67	0.80
	2	1.00	1.00	1.00		3	1.00	0.80	0.89
6	1	1.00	0.19	0.32		4	1.00	1.00	1.00
	2	1.00	0.40	0.57	15	1	1.00	0.14	0.25
	3	1.00	0.67	0.80		2	1.00	0.38	0.55
	4	1.00	1.00	1.00		3	1.00	0.75	0.86
7	1	1.00	0.14	0.25		4	1.00	1.00	1.00
	2	1.00	0.75	0.86	16	1	1.00	0.19	0.32
	3	1.00	1.00	1.00		2	1.00	0.67	0.80
8	1	1.00	0.19	0.32		3	1.00	0.80	0.89
	2	1.00	0.44	0.62		4	1.00	1.00	1.00
	3	1.00	0.67	0.80	17	1	1.00	0.14	0.25
	4	1.00	1.00	1.00		2	1.00	0.50	0.67
9	1	1.00	0.14	0.25		3	1.00	1.00	1.00
	2	1.00	0.75	0.86	18	1	1.00	0.19	0.32
	3	1.00	1.00	1.00		2	1.00	0.44	0.62
10	1	1.00	0.19	0.32		3	1.00	0.67	0.80
	2	1.00	0.44	0.62		4	1.00	1.00	1.00
	3	1.00	0.67	0.80	19	1	1.00	0.14	0.25
	4	1.00	1.00	1.00		2	1.00	0.50	0.67
11	1	1.00	0.14	0.25		3	1.00	1.00	1.00
	2	1.00	0.60	0.75	20	1	1.00	0.19	0.32
	3	1.00	1.00	1.00		2	1.00	0.67	0.80
						3	1.00	1.00	1.00

(c) Across All Users. NumMinSets: The Number of Users who Achieved a 100% F-measure in this Round

Round	Recall	Precision	F-measure	NumMinSets
1	1	0.30	0.42	1
2	1	0.67	0.78	5
3	1	0.83	0.90	6
4	1	0.98	0.99	7
5	1	1	1	1

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

fold, we learn a Bayesian network on the training set consisting of just the score variable and the minimal set of markers together with their corresponding user context and restaurant attributes. Predictions on examples in the test set are then performed using the Netica-J API, via the process described previously in Section 3.2.3.

In each validation fold, prediction accuracies are recorded for the baseline where all values are available, and under scenarios where a context value (and hence its dependent marker value) is missing in the test examples. The BN results are compared against the prediction accuracies from the *J4.8 decision tree* classifier and the *naive bayes* classifier that are implemented in the *Weka* knowledge analysis tool [110], using the default options given in *Weka's Explorer* GUI.

We first investigate if the learned network comprising just the minimal set of markers and the score (the *upper tier* in our model of Figure 3.2) is already sufficient to handle the missing context values. Over five rounds of two-fold cross-validation, we observe the prediction accuracy for score when all the contexts are available, and when the contexts (and hence their dependent markers) go missing one at a time. The detailed results are shown in Table 3.3(a). All the three methods of Bayesian network, decision tree and naive bayes suffer a close to 20% drop in accuracy with missing context values, suggesting that the important contexts and their user-specific interdependencies have to be captured directly during our learning process.

Table 3.3(b) summarizes the accuracies when the important context and restaurant attribute values from the same examples are incorporated. Both decision tree and naive bayes suffer a close to 20% drop in accuracy with missing values, and naive bayes has a much lower accuracy of 92% with complete data, i.e., when no important context value is missing. Bayesian network and decision tree perform equally well when data is complete, but Bayesian network significantly outperforms decision tree and naive bayes to maintain a 99% accuracy even with missing context inputs. The results thus show that the causal dependencies among the context elements are indeed the key to effective compensations by Bayesian network. The fewer markers enable the minimal set of important parameters to be quickly and reliably identified, before the relevant causal dependencies are accurately captured through automatic learning from data.

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

Table 3.3: Average Prediction Accuracy with Missing Context Values (%). Complete: Data is Complete; Missing: With One Context Missing.

(a) Learning and Predicting With Upper Tier.

User	Bayesian Network (%)		Decision Tree (%)		Naïve Bayes (%)	
	Complete	Missing	Complete	Missing	Complete	Missing
1	100	76.8	100	76.8	98.4	76.2
2	100	80.5	100	77.5	96.5	77.7
3	100	70.5	100	66.3	89.6	64.7
4	100	86.2	100	68.6	96.7	76.1
5	100	79.8	100	75.9	100	75.9
6	100	81.2	100	70.4	83.4	75.0
7	100	78.8	100	71.6	85.0	72.2
8	100	83.1	100	83.4	98.3	82.5
9	100	83.9	100	83.9	100	83.9
10	100	80.5	100	77.0	94.9	73.8
11	100	90.1	100	90.1	100	90.1
12	100	86.5	100	83.8	98.3	84.3
13	100	89.6	100	89.6	100	89.6
14	100	84.9	100	77.7	92.7	77.4
15	100	74.9	100	72.8	96.6	75.1
16	100	83.3	100	82.4	84.4	75.9
17	100	89.4	100	84.2	87.4	85.7
18	100	84.0	100	78.7	83.7	74.6
19	100	86.3	100	86.3	96.7	84.7
20	100	86.8	100	78.1	72.7	69.3
Avg.	100	82.9	100	78.8	92.8	78.2

(b) Learning and Predicting With Both Tiers.

User	Bayesian Network (%)		Decision Tree (%)		Naïve Bayes (%)	
	Complete	Missing	Complete	Missing	Complete	Missing
1	100	100	100	78.4	95.0	81.7
2	100	100	100	81.7	88.3	80.1
3	100	100	100	64.5	91.4	74.3
4	100	100	100	77.8	89.5	80.9
5	100	100	100	77.5	92.9	80.9
6	100	100	100	82.9	79.4	76.5
7	100	98.4	100	71.6	86.4	78.1
8	100	99.6	100	83.8	94.8	89.8
9	100	100	100	83.9	100	93.2
10	100	100	100	79.5	94.9	78.7
11	100	98.4	100	93.4	99.6	93.1
12	100	95.4	100	85.4	96.5	84.7
13	100	98.9	100	93.1	100	94.8
14	100	100	100	77.7	87.2	87.6
15	100	94.2	100	76.2	85.3	79.4
16	99.8	99.1	100	86.4	92.5	85.4
17	100	98.4	100	86.5	96.7	89.4
18	100	98.8	100	80.3	86.8	80.7
19	100	98.3	100	88.0	96.8	85.2
20	100	100	100	84.6	88.0	79.6
Avg.	100	99.0	100	81.7	92.1	83.7

3.5.3 Prediction with a Single Erroneous Context

In all our previous experiments, *noise* or erroneous context inputs are absent from the training and test sets. Here in our third set of experiments, we evaluate if the prediction accuracy on *score* can still remain high despite the errors in important context values. For each of the users modelled in Table 3.1, we have prepared two sets of one-thousand examples for n rounds of two-fold cross-validation, where n is the number of contexts that are important for each user. In each round, we simulate the scenario where one of these n contexts significant to the user has errors, but the system does not know which is the erroneous context nor what is its true error rate. The examples for each user are generated in the same manner as we described in Section 3.4 and the start of Section 3.5.1; for each error rate, a thousand observations that are injected with errors are used for learning, and another set of a thousand unseen test cases carrying similar errors are used to validate the resulting prediction accuracy on the score variable.

Within each fold, we first learn a Bayesian network on the training set comprising just the set of markers and the score. Through a stepwise elimination process as described in Section 3.5.1, we identify the minimal set of markers important to each user. The optimal set of learning parameters then consists of the score variable, the important markers, as well as their corresponding user context and restaurant attributes. With this information in hand, we can now evaluate our proposed procedure (Procedure 3) in Section 3.3.2 for handling a single unidentified erroneous context suffering from an unknown rate of error.

With reference to the user models described in Table 3.1, our observations for User 1 are summarized in Table 3.4. As before, we compare the average prediction accuracies for our Bayesian network-based approach to the results achieved with decision tree and naive bayes.

The three methods exhibit comparable prediction accuracies when we treat the erroneous values as specific evidence (results not shown). However, Table 3.4 shows that using Bayesian network that takes into account the error rates in contexts consistently gives significantly better results than the other methods on the same test set. This confirms that our emphasis on discovering error rates for prediction is indeed sound. Furthermore, results obtained when we repeat the learning on cleansed data show that Bayesian networks trained on cleansed data sets perform better than their counterparts trained on uncleansed data sets on all occasions.

Table 3.4: Average Prediction Accuracy for User 1 (%). Error Rate of 30% is Used in All Experiments; Err-Ctx: Context in Error; DT: J4.8; NB: Naive Bayes; BN: Bayesian Network, Using Likelihood Estimates.

Fold	Err-Ctx	Without Data Cleansing			With Data Cleansing		
		DT	NB	BN	DT	NB	BN
1	UP1	86.03	84.51	91.74	86.03	87.03	100.0
	UP3	79.55	79.84	89.25	79.55	81.67	100.0
	US4	97.87	94.08	98.45	100.0	94.95	100.0
	Average	87.82	86.14	93.15	88.53	87.88	100.0
2	UP1	85.06	85.73	94.01	85.06	86.97	100.0
	UP3	78.05	79.67	88.76	78.05	81.29	100.0
	US4	98.09	93.50	99.61	100.0	94.40	100.0
	Average	87.07	86.30	94.13	87.70	87.55	100.0

The charts in Figure 3.7 compare the average prediction accuracies of Bayesian network against decision tree and naive bayes over all modelled users when one context is in error at a time. Even at the high error rate of 45%, Bayesian network maintains its accurate predictions, unlike the other methods. This is because our proposed procedure has reliably discovered the erroneous inputs and presented their values to the BN as likelihood estimates during inferences, such that these could then be compensated by the other presented input values. The strength of our approach thus stems from its ability to reliably capture the uncertainty in the erroneous context values in its estimation of their likelihoods. Baselines consisting of decision tree and naive bayes classifiers learned from the original uncleansed data are added for comparison. The results show that all the methods benefit from cleansing, and that by using the discovered training errors on a context as an estimate for its rate of error, our proposed procedure effectively derives its likelihoods to achieve higher accuracy on erroneous data.

3.5.4 Performance under Small Number of Training Examples

We now evaluate the performance of our proposed error-handling procedure from Procedure 3 under scenarios where we only possess a smaller number of training examples. Specifically, we compare the average prediction accuracy of Bayesian network against J4.8 decision tree and naive bayes classifier when an important context suffers from various rates of error. For each of our modelled users, we reserve the last hundred of their examples for testing. In each trial,

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

3.7.a: BN versus DT. Baseline is DT Learned on Uncleansed Data.

3.7.b: BN versus NB. Baseline is NB Learned on Uncleansed Data.

Figure 3.7: Average Prediction Accuracy with A Single Context In Error.

we randomly select k of the remaining examples for training, where k ranges from ten to sixty examples. We present the average results from ten such trials in Figure 3.8 and Figure 3.9.

With reference to these figures, the advantage of our approach based on the Bayesian network is related to the number of training examples. Using a statistical paired sample t -test [109] for difference in mean values, we confirm that our method significantly outperforms the J4.8 decision tree at the 0.05 level when there are at least forty training examples, and that we outperform the decision tree at the 0.01 level given fifty or more examples. Our proposed method also outperforms the naive bayes classifier at the 0.01 level when there are as few as twenty

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

3.8.a: Error Rate 0%.

3.8.b: Error Rate 10%.

3.8.c: Error Rate 20%.

3.8.d: Error Rate 30%.

3.8.e: Error Rate 40%.

3.8.f: Error Rate 50%.

Figure 3.8: Average Prediction Accuracy as the Context Error Rate Varies. Baseline is Decision Tree (DT) Learned on Uncleansed Data.

training examples, and we get an increasingly superior performance beyond twenty examples. Against the decision tree, the Bayesian network learning is inferior only when the training set is restricted to a size of thirty or fewer examples; against the naive bayes, a training set of merely ten examples is insufficient for Bayesian network learning. This observation is consistent with the *CaMML*'s authors expectation that *CaMML* would not perform as well as simpler machine learning methods like decision tree and naive bayes when the amount of data is so small that it is unable to find the correct causal model during its search through the model space [111].

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

3.9.a: Error Rate 0%.

3.9.b: Error Rate 10%.

3.9.c: Error Rate 20%.

3.9.d: Error Rate 30%.

3.9.e: Error Rate 40%.

3.9.f: Error Rate 50%.

Figure 3.9: Average Prediction Accuracy as the Context Error Rate Varies. Baseline is Naive Bayes (NB) Learned on Uncleansed Data.

Our results show that when there are erroneous context inputs, Bayesian network outperforms the competing methods as more examples become available. This is a significant advantage because within typical context-aware recommender environment such in mobile computing, erroneous context inputs are the norm. It is hence important that our method effectively overcomes these after prolonged usage. Earlier results (Figure 3.7) have shown that with ample examples, our method significantly outperforms both decision tree and naive bayes classifier.

It is interesting to note that when the number of training examples is small, data cleansing

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

appears to have a negative effect on J4.8 decision tree and naive bayes in these experiments. With reference to the plots in Figure 3.8 and Figure 3.9, it appears that any further elimination of suspicious examples from an originally small set of training examples can in fact make it harder to discover meaningful patterns. However, we expect data cleansing to have a positive effect on performance given sufficient training examples, as evident from the convergent trends for lines “DT” and “DT (baseline)” and lines “NB” and “NB (baseline)” as the number of training examples goes up to sixty. This is in agreement with the earlier results from Section 3.5.3, where the benefits from data cleansing with ample training examples are clearly demonstrated.

Our approach outperforms the other methods as the number of training examples increases. However, Bayesian network learning using the *CaMML* is less effective when the number of training examples is so few that the correct causal model cannot be identified [111]. One usual solution is to request the user to label a small number of, say thirty or forty, examples initially. Other possible solutions include using various machine-learning techniques to maximize the utility from smaller data sets, through increasing the volume of training examples by labelling unlabelled examples using available labels as a guide. One popular approach, *semi-supervised learning* [112], selectively labels the unlabelled (unrated) examples to augment the labelled points, for example through the use of statistical Expectation Maximization (EM). One other popular approach that can be explored is that of *active learning* or *sample selection*, in which training examples are acquired incrementally and the system attempts to use what it has already learned to select only those most informative new examples for the user to rate [113]. Specific techniques for applying each of these approaches have been developed, and these techniques have been shown to significantly reduce the labelled examples required for real-world problems [83]. As techniques such as semi-supervised learning may introduce some errors in the training labels, more experiments are necessary to validate their effectiveness in practice. This is an interesting area that involves many other issues which are outside the scope of this Thesis.

3.5.5 Prediction with Multiple Erroneous Contexts

In previous experiments with erroneous context, we assume the knowledge that just one context is in error. To validate our procedure where there is an unknown number of error contexts, we

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

Table 3.5: User Preference Rule and Model for User 21.

User	Preference Rule	Causal Model
21	M3, M5, M10, M11, M12, M14, M15, M21	Model 21

Table 3.6: Results of Error Discovery for User 21. Err-Ctx Added: Error Context Added in This Step/Iteration; Err-Ctxs Found: The Error Contexts Found So Far; Err-Rate Ests: Estimated Error Rates for Use in Next Step.

Fold	Step	Err-Ctx Added	Err-Ctxs Found	Err-Rate Ests
1	0	UP5	UP5	0.321
	1	UP3	UP5, UP3	0.321, 0.212
	2	UP2	UP5, UP3, UP2	0.321, 0.212, 0.099
	3	-	UP5, UP3, UP2	0.319, 0.210, 0.099
2	0	UP5	UP5	0.307
	1	UP3	UP5, UP3	0.307, 0.201
	2	UP2	UP5, UP3, UP2	0.298, 0.201, 0.09
	3	-	UP5, UP3, UP2	0.298, 0.201, 0.09

simulate a user with eight important markers (see Table 3.5). Similar to the setup described in Section 3.5.3, we generate two thousand examples for this user according to her preference rule and causal model (her preference markers are listed in Table 3.5, while the causal model is presented in Appendix B as Causal Model 21). We randomly partition these examples into two equal sets for a two-fold cross-validation, with three contexts each having some error rate.

Following the steps described in our minimal set learning Procedure 1 of Section 3.2, the correct minimal set of markers is identified in both the validation folds. Next, applying Procedure 4 of Section 3.3.3 to the data with errors, we are able to identify all the three error contexts and deduce close estimates of their error rates. The evolution of the model at each step of the error-discovery process is traced in Table 3.6. Error discovery terminates when the observed error rate of the next most erroneous context falls to 0.0.

Table 3.7 compares the performance when just the misses for UP5 (the “most-erroneous” context) are removed, when misses for either UP5 or UP3 (the two most-erroneous contexts) are removed, and also when the misses for all three identified error contexts of UP5, UP3 and UP2 are removed from the training set. The prediction performance for J4.8 decision tree is added for comparison. The results confirm that test prediction accuracy increases steadily

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

Table 3.7: Prediction Performance for User 21. Ctxs_for_Cleansing: Contexts Based On Which Misses Are Removed; Avg. Acc. (%): Average Accuracy on Score; BN: Bayesian Network, Using Likelihood Estimates; DT: J4.8 Decision Tree.

Ctxs_for_Cleansing	Avg. Acc. (%) for BN	Avg. Acc. (%) for DT
UP5	92.44	69.81
UP5, UP3	98.12	71.92
UP5, UP3, UP2	100.0	80.17

3.10.a: With 2 Error Contexts.

3.10.b: With 3 Error Contexts.

Figure 3.10: Average Prediction Accuracy when there are Multiple Error Contexts. Baseline is DT Learned on Uncleansed Data.

as suspicious examples are cleansed from the training set. The mean error is significant at the 0.05 level (one-tailed, i.e., whether mean error is larger than 0.0) only for the first step where no example has yet been removed. As such, the significance test suggests to remove just the most-erroneous context (UP5) to arrive at the final model. From Table 3.7, this yields an average test prediction accuracy exceeding 90%, compared to just around 70% for decision tree classifier.

Figure 3.10 and Figure 3.11 compares the prediction performance of the methods over all

3.11.a: With 2 Error Contexts.

3.11.b: With 3 Error Contexts.

Figure 3.11: Average Prediction Accuracy when there are Multiple Error Contexts. Baseline is NB Learned on Uncleansed Data.

Table 3.8: Average Prediction Accuracy of Various Classification Algorithms on the Goats Data Set.

Accuracy (%)				
N Neigh	ID3	Percept	BackP	PEBLS
62.0	64.7	66.3	67.0	62.7
Naive Bayes	Rocchio	J4.8	BN	
62.9	69.4	61.0	64.8	

the modelled users described in Table 3.1. Figure 3.10.a and Figure 3.11.a present the results when two contexts (unknown) are in error, while Figure 3.10.b and Figure 3.11.b present the results when there are three erroneous contexts. As expected with the larger proportion of contexts in error, prediction accuracy deteriorates across all the methods. As per our observations from Section 3.5.3, all the methods benefit from data cleansing, albeit this benefit reduces as more contexts are in error. These results suggest that, by accounting for the uncertainty in the evidence, our procedure can indeed outperform the other methods when multiple context parameters are suffering from errors at rates which are not revealed to any of the learning methods.

3.6 Further Validation: Personalized Web Page Recommendation

In this section, we extend our experiments to a real user recommendation data set known as the Syskill and Webert Web Page Ratings data set [114] (available from UCI). This data comprises of the HTML sources of web pages and their corresponding ratings (“hot”, “medium/cold”) from a real user. Seventy web pages on the subject of “Goats” are used. The attributes are English words that are extracted from these web pages, excluding those words from the standard SMART list of stopwords [115]. The original study [116] has reported the average prediction accuracy, i.e., percentage of misclassified test examples, for algorithms including naive bayesian [117], nearest neighbor [117], perceptron nets [118] and ID3 decision trees [119]. Based on this real data set, we explore if causal dependencies among word attributes can be effectively discovered and exploited by our proposed Bayesian network-based mechanism for predicting the user ratings of web pages.

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

Figure 3.12: Bayesian Networks Learned from Goats Web Page Data Set for a Real User.

Following the setup described in [116], we train J4.8 and Bayesian network on twenty randomly-selected examples and test on the rest. The average prediction accuracies over ten trials obtained by the J4.8 DT and the Bayesian network are presented in Table 3.8 together with the previously reported results. In each trial, the top-k most informative words are selected using the same information gain criterion as in [116]. However, because Netica-J API [101] has a hard-coded restriction on the network size, only the top-32 most informative words are used for Bayesian network whilst 96 word attributes are used for the others. As highlighted by [116], using fewer attributes can cause problems for the Bayesian network because important discriminating attributes may be excluded. Despite this, by employing our proposed minimal set learning Procedure 2, we are able to learn Bayesian networks that outperform on average many of the other algorithms including naive bayes and J4.8. This shows that the minimal set that our iterative approach has automatically identified is effective in predicting the real user's rating, although for the small training sample size, BN learning using the search-based *CaMML* tool is not the best approach to predict with perfect data. However, like the decision tree and naive bayes that we see so far, the other classification methods do not encode variable dependencies, and would not be as robust as our BN-based approach for handling missing and error values.

Next, training on sixty randomly-chosen examples, we discover the minimal set of important attributes for that user using our iterative learning Procedure 2. Figure 3.12.a shows the network that is learned from the 32 most informative words in a trial. Our procedure has retained the “dairy”, “farm”, and “development” attributes for a second round of learning, in which all these three attributes have remained connected to *score*. As such, they constitute the

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

3.13.a: BN Against J4.8 Decision Tree.

3.13.b: BN Against Naive Bayes Classifier.

Figure 3.13: Average Prediction Accuracy When There Are Missing Values.

minimal set of important attributes. Although markers are unavailable, our two-tiered context model can effectively capture the causal dependencies by including attributes that are connected to the minimal set (“doe”, “production”, “management”, “break”, “return”). The final learned network is given in Figure 3.12.b. The proposed approaches for handling missing and erroneous values that we have evaluated previously with the restaurant recommender data can now be applied to this problem.

Figure 3.13 compares the prediction accuracies of Bayesian network against J4.8 decision tree and naive bayes for various rates of missing values in the test sets. In the experiments, only one attribute from among the minimal context set is missing at a time. A missing rate of $x\%$ implies that the problematic attribute has a $x\%$ probability of missing from an example. The results show that Bayesian network consistently outperforms the other methods in overcoming missing inputs to predict more accurately on the test sets. Figure 3.14 compares the performance when various probabilities of errors are introduced. Specifically, in each of the ten trials, a randomly-chosen attribute from the minimal set is injected with errors. We apply

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

Table 3.7: Prediction Performance for User 21. Ctxs_for_Cleansing: Contexts Based On Which Misses Are Removed; Avg. Acc. (%): Average Accuracy on Score; BN: Bayesian Network, Using Likelihood Estimates; DT: J4.8 Decision Tree.

Ctxs_for_Cleansing	Avg. Acc. (%) for BN	Avg. Acc. (%) for DT
UP5	92.44	69.81
UP5, UP3	98.12	71.92
UP5, UP3, UP2	100.0	80.17

3.10.a: With 2 Error Contexts.

3.10.b: With 3 Error Contexts.

Figure 3.10: Average Prediction Accuracy when there are Multiple Error Contexts. Baseline is DT Learned on Uncleansed Data.

as suspicious examples are cleansed from the training set. The mean error is significant at the 0.05 level (one-tailed, i.e., whether mean error is larger than 0.0) only for the first step where no example has yet been removed. As such, the significance test suggests to remove just the most-erroneous context (UP5) to arrive at the final model. From Table 3.7, this yields an average test prediction accuracy exceeding 90%, compared to just around 70% for decision tree classifier.

Figure 3.10 and Figure 3.11 compares the prediction performance of the methods over all

3.11.a: With 2 Error Contexts.

3.11.b: With 3 Error Contexts.

Figure 3.11: Average Prediction Accuracy when there are Multiple Error Contexts. Baseline is NB Learned on Uncleansed Data.

CHAPTER 3. LEARNING THE MINIMAL SET OF CONTEXTS AND EXPLOITING THE CONTEXT DEPENDENCIES FOR ROBUST RECOMMENDATIONS

approach reliably identifies the minimal set of important input parameters for a particular real user, and also effectively discovers and exploits the dependencies among these parameters to produce a recommender that is robust against both missing and erroneous input values.

3.7 Chapter Summary

Acquisition of contexts poses unique challenges to context-aware recommender systems, in particular those operating in the volatile mobile environment. The limited resources in these systems make minimizing context acquisition a practical need, and the uncertainty in the environment makes missing and erroneous context inputs a major concern. In this chapter, we present our Bayesian network (BN)-based approach for building recommender systems that minimize context acquisition and predict robustly under noisy conditions.

The proposed learning approach iteratively trims the BN-based context model until it contains just the minimal set of context parameters that are important to a user. Using the platform of *context-aware content-based recommender systems*, we demonstrate that a two-tiered context model enables the effective learning of context dependencies from data using an existing BN learning tool, and that these captured dependencies equip a recommender system to compensate readily for missing context inputs during recommendations. To enable the recommender systems to handle erroneous context inputs, we have devised a novel and effective procedure that reliably discovers which are the erroneous contexts, estimates their individual rate of error, and automatically compensates for the uncertainty in those erroneous contexts' values during the BN recommendation process.

We validate the proposed techniques on a restaurant recommendation data set and a web page recommendation data set. Empirical results from both problems show that the minimal set of contexts is reliably discovered for each user, and that the learned BN consistently outperforms J4.8 decision tree (DT) and naive bayes (NB) classifier when important contexts are missing. Furthermore, the proposed error-handling procedure enables BN to outperform both J4.8 DT and NB when presented with erroneous context inputs, to generate recommendations that are significantly more robust.

Chapter 4

Explaining Recommendations in Bayesian Networks

As described in Section 2.2 of Chapter 2, Bayesian network (BN) represents causal dependencies as directed arcs, and derives belief values by multiplying conditional probabilities between variables. In the previous chapter, we witnessed that these captured dependencies enable BN to predict accurately even when important inputs are missing or in error. To gain user acceptance, however, recommendations from the BN have to be understandable for users. As the domain becomes more complex, the gap between user and system knowledge widens, and the need for explanation increases. Indeed, past studies have shown that a prominent requirement by human users of recommendation systems is the ability to *explain* their inferences [121, 122]. Since BN infers based on the probabilistic interactions amongst its variables, the explanations should reflect these interactions and the resulting compensations of missing and erroneous values.

In this chapter, we introduce the **Explaining BN Inferences (EBI)** procedure to address the above problem. The proposed EBI procedure consists of three key steps: First, EBI restructures the BN (without modifying its joint distribution) such that the target has its Markov nodes as parents. Next, EBI condenses the target's table of conditional probabilities into decision trees (DT) [123]. Finally, those explanatory nodes corresponding to specific contexts are identified by traversing the DTs, and the corresponding EBI explanations are generated dynamically.

The rest of the chapter is organized as follow. In Section 4.1, we present the preliminaries of our work, including the problem background, key concepts on BN and variable independencies, and the limitations in the existing solutions. This is followed by the detailed discussion of our EBI procedure in Section 4.2 and Section 4.3, and its experimental validation in Section 4.4.

Figure 4.1: A Bayesian Network (BN) for Movie-Watching.

4.1 Preliminaries and Related Work

4.1.1 Background of Our Work

Existing methods generally explain in terms of a set of input values that generates a target distribution similar to the inferred distribution. Figure 4.1 shows a BN for movie-watching. Suppose the boolean input values are given as $\{workDone = \text{true}, hasMoney = \text{true}, hasFriend = \text{true}\}$. Methods like INSITE [124] and BANTER [125] will check if individual input influences the target significantly, then search for the most influential path from each significant input to the target. An explanation for *watchMovie* may be:

Before presenting any evidence, the probability of *watchMovie* is p_a .

The following findings are considered important (in order of importance):

- *workDone* results in a posterior probability of p_{b_1} for *watchMovie*.
- *hasMoney* results in a posterior probability of p_{b_2} for *watchMovie*.
- *hasFriend* results in a posterior probability of p_{b_3} for *watchMovie*.

Their influences flow along the following paths:

- *workDone* influences *hasTime*, which influences *watchMovie*.
- *hasMoney* influences *watchMovie*.
- *hasFriend* influences *watchMovie*.

Presenting the evidence results in a posterior probability of p_b for *watchMovie*.

Such explanations conceal the interactions among the parents into *watchMovie*, thus disregarding the fact that inferences for *watchMovie* depend strongly upon its parents' interactions. In addition, there is no mechanism to explain missing and erroneous inputs.

In contrast to existing approaches that explain inferences in terms of individual evidence variables, we propose to explain the BN's inferences on a target using a conjunction of the values in its *Markov blanket* [17], which includes the target's parents, children, and the children's other parents. In the movie example, we prefer to explain *watchMovie* in terms of the interactions among *hasTime*, *hasMoney* and *hasFriend*, rather than the inputs individually. Now, suppose

that the *watchMovie*'s probabilities are independent of *hasFriend* when a user *hasTime* and *hasMoney*. This is an instance of context-specific independence (CSI) [126]. We can exploit this to generate a more succinct explanation:

BN predicts *watchMovie* is true with probability p_b because

hasTime is true with probability p_x , and

hasMoney is true with probability p_y .

Such *probabilistic causal explanations* [127, 128] take the form of *probabilistic rules* [129]:

$$\text{if } \langle \text{conditions} \rangle \text{ then } \langle \text{conclusion} \rangle \text{ with probability } \langle p \rangle \quad (\text{Eq. 4.1})$$

where p is given by the probability $P(\text{conclusion}|\text{conditions})$.

The proposed **Explaining BN Inferences (EBI)** procedure is unique in that:

- Whereas most existing methods restrict their explanation to the evidence nodes, EBI explains the target node's value using nodes in its Markov blanket that shield it from the rest of the network. This suits our human inclinations to reason with a few specific heuristics [130], and to better accept causal stories as explanations [131].
- When observations are presented as evidence for BN inference, some of the values may be incorrect. While some explanation methods highlight erroneous values as conflicting findings, no existing method can explain how errors are compensated during inference. Also, current methods do not explain how missing values along the influence paths to the target are derived. In contrast, EBI provides users with concise insights into these compensations, presented as layers of explanatory rules.

4.1.2 Bayesian Network and Variable Independencies

A Bayesian network (BN) is a directed acyclic graph, wherein each node represents a random variable and an edge indicates direct dependency between two variables. By its structure and conditional probabilities, a BN encodes a joint distribution that describes the domain's probabilistic semantics. We shall now discuss some specific key concepts of the BN that are needed for a clear appreciation of our proposed BN explanation approach.

CHAPTER 4. EXPLAINING RECOMMENDATIONS IN BAYESIAN NETWORKS

4.2.a: Example BN.

X_3	X_4	X_5	$P(X_6=0 \mid X_3, X_4, X_5)$	$P(X_6=1 \mid X_3, X_4, X_5)$
0	0	0	p1	1 - p1
0	0	1	p1	1 - p1
0	1	0	p1	1 - p1
0	1	1	p1	1 - p1
1	0	0	p2	1 - p2
1	0	1	p2	1 - p2
1	1	0	p3	1 - p3
1	1	1	p3	1 - p3

4.2.b: CPT of Node X_6 .

Figure 4.2: Example Bayesian Network (BN) and Conditional Probability Table (CPT).

Definition (Conditional Independence): A variable X is conditionally independent of non-descendants (denoted hereafter as $NDes(X)$) given its parents' values [17]:

$$P(X \mid Parents(X), NDes(X)) = P(X \mid Parents(X)) \quad (\text{Eq. 4.2})$$

The example BN in Figure 4.2.a involves binary variables X_1 to X_6 , ordered such that non-descendants of X_i are labelled with indices less than i . The BN's joint probability, denoted as $P(X_1 = x_1, X_2 = x_2, \dots, X_6 = x_6)$, or $P(x_1, x_2, \dots, x_6)$, can be factorized as:

$$P(x_1, x_2, \dots, x_6) = P(x_1) \times P(x_2 \mid x_1) \times \dots \times P(x_6 \mid x_1, \dots, x_5) = \prod_i P(x_i \mid x_1, \dots, x_{i-1}) \quad (\text{Eq. 4.3})$$

By exploiting conditional independence (Eq. 4.2), the joint probability reduces to

$$P(x_1, x_2, \dots, x_6) = \prod_i P(x_i \mid Parents(x_i)), \quad Parents(x_i) \subseteq \{x_1, \dots, x_{i-1}\} \quad (\text{Eq. 4.4})$$

This allows joint probability to be defined as a product of *conditional probability tables*.

Definition (Conditional Probability Table): The conditional probability table (CPT) of a node X tabulates its distribution across value assignments to its parents [17].

For example, the CPT in Figure 4.2.b tabulates the conditional distribution of X_6 . In the CPT, $P(X_6 = 0 \mid X_3 = 0, X_4 = 0, X_5 = 0)$ is given by conditional probability p1.

In many practical applications [126, 132–134], the CPTs can be further decomposed for faster inferences if the target value is independent of certain parents given that certain other parents have some values. Such a regularity, which is essentially an independence relation that holds only under specific value assignments to certain variables, is known as a *context-specific independence* (CSI). A formal definition is given by Boutilier et al. [126]:

Definition (Context-Specific Independence): Let X , Y , Z and C be pairwise disjoint sets of variables. Sets X and Z are *context-specifically independent*, or *contextually-independent*, given Y and the *context* $c \in \text{val}(C)$, if the conditional probability $P(X \mid Y, c, Z) = P(X \mid Y, c)$ whenever $P(Y, c, Z) > 0$.

For the CPT in Figure 4.2.b, $P(X_6 = 0 \mid X_3 = 0, X_4, X_5)$ is p1 regardless of the values of X_4 and X_5 , i.e., X_6 is *contextually independent* of X_4 and X_5 given $(X_3 = 0)$.

Conditional independence implies that given the parent values of a target node with no descendant, the target is independent of other nodes. However, this level of information is insufficient to “block” a variable from its descendants, because we can only assert that

$$P(X \mid NDes(X), Des(X)) = P(X \mid Parents(X), Des(X)) \quad (\text{Eq. 4.5})$$

where $Des(X_i)$ is the set $\{X_{i+1}, X_{i+2}, \dots\}$ of X_i ’s descendants. Completely predicting a node’s behavior requires a knowledge of the larger set of nodes in its *Markov blanket* [17].

Definition (Markov Blanket): Markov blanket of X is the set of nodes $MB(X)$ comprising of X ’s parents, children, and spouses (the child nodes’ other parents) [17].

Definition (Markov Node, Markov Value): The nodes in X ’s Markov blanket constitute the *Markov nodes* of X ; the Markov nodes’ values are node X ’s *Markov values*.

For example, node X_5 in Figure 4.2.a has Markov nodes X_2 , X_6 , X_3 and X_4 (their values are the Markov values of X_5). The *Markov property* asserts that direct dependencies must be shown by arcs, i.e., there cannot be “backdoors” in a BN [65]. Every other node is thus independent of or *d-separated* from a node X when conditioned on $MB(X)$. Formally,

$$P(X \mid MB(X), MB'(X)) = P(X \mid MB(X)), \quad MB'(X) \cap (MB(X) \cup X) = \emptyset \quad (\text{Eq. 4.6})$$

where $MB'(X)$ comprises the set of remaining variables in the Bayesian network that do not fall within the Markov blanket of node X , and node X is itself not a part of $MB'(X)$.

During an inference, the belief or posterior probability of each of the variables’ values is updated according to the evidence presented to the BN. Bounded by a common joint distribution, the values that are assigned to the target variable’s Markov nodes completely explain the target variable’s inferred value, because by just having the Markov values, the most-probable target value is consistent with its expected value under the given evidence.

4.1.3 Related Work on Explaining BN Inferences

Existing tools generally display BN inferences using bars, meters, or other graphical expressions of the differences in probabilities among variable values. Despite the easier interpretation, they are limited to examining single variable's distributions, and it remains hard to comprehend the overall inference process. No explanation method has attempted to control the level of details so as to cater to the user's knowledge, or how inquisitive a user is. There is no dialogue to allow a user to gather explanations progressively [135].

INSITE [124] and BANTER [125] are well-recognized BN explanation approaches. Both methods identify the individually influential input nodes and the paths along which their influences upon the target flow. The difference is that BANTER *instantiates inputs individually*, whilst INSITE *removes input values individually* to compare their effects on the target's probabilities. Haddawy et al. [125] highlight that both BANTER and INSITE cannot detect influential inputs that interact among themselves, since their exhaustive approach would require an exponential number of network evaluations in order to do so.

Chajewska and Draper [136] reduce this computational complexity by searching for explanations that are good but not necessarily the best. They propose postulates for the measure of explanation quality and provide examples of such functions. Their algorithm searches for explanations that satisfy a quality threshold. It remains a challenge to determine whether there exists an explanation with a quality exceeding a specific threshold, since this would still require an examination of all possible subsets of evidence variables.

Zhou et al. [137] propose the extraction of classification and correlation rules based on the target's probabilities. The rule set is a classification model that is compliant with the BN. However, because they do not have a mechanism to isolate the influential variables from among all the input values, their explanations can only be in terms of all the inputs. Moreover, because both the order of inputs and their relations under specific contexts are never considered, any interactions among inputs are neglected by the resulting rules.

The above approaches assume that the user prefers explanations which connect the evidence variables to the target. They suffer when the assumption fails for two reasons:

- Firstly, the number of variables is often overwhelming. For example, there are usually thousands of variables in domains such as biomedicine, text and multimedia recommendations. As a typical user understands only a handful of these, it is not always meaningful to explain inferences using variables that are part of the evidence.
- Secondly, the aforementioned approaches require an exponential number of analyses for all possible subsets of evidence variables. They are limited to analyzing findings individually, and are computationally intractable for reflecting variable interactions.

Scenario-based reasoning [138] argues that the inferences can instead be explained by inputs that are “relevant” to the target, even though they may not be part of the evidence. Firstly, the relevant variables are selected based on the criterion of *d-separation* [17]. Each combination of values for these variables constitutes one possible scenario, or causal story. As there may be a large number of scenarios, only the most probable ones are presented.

Similar reviews on BN explanation approaches can be found in [135] and [139]. To the best of our knowledge, no prior work has exploited the variable independencies in BN to isolate the target’s Markov values for explaining its inferences. In addition, no prior work has exploited variable interactions to simplify the explanations, and none of the existing approaches can explain the underlying BN compensations during inferences.

4.2 Our Approach to Explaining BN Inferences

We introduce the Explaining BN Inferences (EBI) procedure, as summarized in Figure 4.3. Recall from Section 4.1.2 that the Markov blanket is the minimal set of nodes that completely predicts the behavior of the target node. EBI first restructures the local dependencies around the target node through arc reversals, in such a way that its Markov nodes become its parents while maintaining the same joint distribution. The resulting conditional probability table (CPT) of the target node describes its probability distribution over all combinations of Markov values. To reduce the complexity of the rules that are generated to explain the inferences, EBI exploits context-specific independence (CSI) and explains in terms of only the Markov values that are significant. From the CPT, a decision tree (DT) [123] that captures the CSI among the Markov

CHAPTER 4. EXPLAINING RECOMMENDATIONS IN BAYESIAN NETWORKS

Figure 4.3: Overview of Our Proposed Explanation Procedure, EBI.

Figure 4.4: Running Example for Section 4.2. Context-Specific Independencies For Target Node E are Captured As Regularities in the Conditional Probabilities of Figure 4.4.c.

nodes is learned for each target value. During recommendation, EBI compares the assigned Markov values against the DT corresponding to the inferred target value, and explains using nodes on the path that matches the current context. In the rest of this section, we elaborate on each of these key component processes of EBI with Figure 4.4 as a running example.

4.2.1 Transforming the BN through Arc Reversals

Given a BN, we can reverse the arc from the target to each of its children, while readjusting their local structures and conditional probability tables such that the BN produces identical inferences as before. In the process, arcs are added from the other parents of each child to the

target. Effectively, all the nodes in the Markov blanket of the target become its parents, which allows us to apply DT induction to the resulting target CPT.

Theorem (Arc Reversal): Arc (i,j) linking node i to node j can be replaced by an arc (j,i) , provided that there is no other directed (i,j) -path in the BN. After the reversal, both nodes inherit each other's conditional predecessors (the proof is given in [140]).

The requirement on the absence of alternative (i,j) -path is necessary and sufficient to ensure that no cycle is created by the arc reversal transformation. For our purpose of reversing the arcs from the target t to each of its child nodes, this requirement is satisfied by sequencing the arc reversals: for a child c with an alternative directed (t,c) -path, the arc that links t to the child node on the alternative path is reversed before the arc (t,c) .

We illustrate the arc reversal transformation [140, 141] with our running example. Figure 4.4.a shows the original BN, and Figure 4.4.b shows the same BN after the arc from node E to node C has been reversed. Before reversal, $Parents(E) - Parents(C) = \{A, B\}$ and $Parents(C) - Parents(E) - \{E\} = \{D\}$. The Markov blanket of E comprises parents $\{A, B\}$, child C , and the other parent of C , namely D . The reversal of $arc(E, C)$ involves setting $Parents(E) - \{C\} = Parents(C) = \{A, B\} \cup \{D\}$ so that E and C share a set of conditioning nodes, and computing their conditional probabilities as follows:

$$P(C|A, B, D) = \sum_E P(E|A, B)P(C|E, D) \quad (\text{Eq. 4.7})$$

$$P(E|C, A, B, D) = \frac{P(E|A, B)P(C|E, D)}{P(C|A, B, D)} \quad (\text{Eq. 4.8})$$

Eq. 4.7 and Eq. 4.8 ensure that the joint distribution of the BN remains unchanged. Eq. 4.7 computes $P(C|A, B, D)$ by summing out or marginalizing the node E from the joint distribution before arc reversal. Eq. 4.8 computes $P(E|C, A, B, D)$ by equating the joint distributions before and after arc reversal, and dividing the joint distribution on each side by $P(C|A, B, D)$. The other child nodes of E and C are unaffected. Figure 4.4 shows that all the nodes in E 's Markov blanket are transformed to become parents of E .

4.2.2 Learning Context-Specific Independencies among Markov Nodes

After arc reversal, the context-specific independencies (CSI) among the Markov nodes are learned from the resulting target conditional probability table (CPT). The CPT entries are

Figure 4.5: Decision Tree (DT) Induced From the CPT Entries for Value e in Figure 4.4.c.

grouped by the target values, and a decision tree (DT) [123] is learned from each group with the conditional probabilities as class labels. For the CPT in Figure 4.4.c, the Markov values and probabilities in the first two columns constitute the group of sixteen entries for value e of node E . Figure 4.5 shows the DT that is learned from this group.

DT induction is suitable as EBI's learning method as it is fast and straightforward. More importantly, each DT contains an *exhaustive* set of *mutually exclusive* paths. These properties guarantee that in the learned DT, there is always a unique path that matches the current Markov value combination. Nevertheless, the quality of the DT induced from the CPT affects the extent to which the context-specific independence among the Markov nodes can be extracted. A discussion about the representation of CPTs by DTs is given in [126]. Assuming that the CPTs are not revised incrementally by incoming examples, the entire process of reversing the arcs and learning the DTs can be completed off-line during preprocessing. The DTs are then referred during recommendation to identify the minimal set of Markov nodes that concisely explains the inference in the current context.

4.2.3 Generating Context-Specific Explanations using Markov Nodes

When the BN predicts a value for the target, EBI dynamically generates an explanation for the inference in the current context. From the DT that corresponds to the predicted target value, EBI selects the path that matches the target's assigned Markov values, and forms the explanation with the nodes on this path. The most time-consuming task during explanation is the identification of the matching path. Zhang [142] points out that the matching operation is more efficient with trees of paths than with sets of rules. Letting n be the number of Markov values that constitute the current context of the target node, the worst-case time complexity

of generating the EBI explanation is merely $O(n)$. We discuss in the next section how EBI is applied recursively to explain compensations during inference; using trees to capture CSI makes EBI efficient even in recursive application.

Consider the running example in Figure 4.4. Given the evidence, the BN recommends a value for target E and similarly for each of its Markov nodes A , B , C and D . Without using the DTs, we can explain the inference on E by matching the sixteen entries (rules) in its CPT to the Markov values. For instance, if the Markov values are $\bar{a}$, b , c and $\bar{d}$, respectively, an explanation can be formulated from the matching CPT entry in row 10 of Figure 4.4.c. If $p_3=0.7$, BN infers the value of E as e with the following explanation:

- E is e with probability 0.7 because A is $\bar{a}$, B is b , C is c , and D is $\bar{d}$.

Instead of matching on the CPT entries, EBI traverses the DT shown in Figure 4.5 to find the path that contains the important Markov values within the current context. The matching process descends from the tree root to a leaf node by following branches compatible with the Markov values. For the given Markov values of $\bar{a}$, b , c and $\bar{d}$, the resulting EBI explanation for $P(e)$ is independent of B and D in the context ($A=\bar{a}$, $C=c$):

- E is e with probability 0.7 because A is $\bar{a}$, and C is c .

Correspondingly, the EBI explanation for $E=\bar{e}$ is given as:

- E is $\bar{e}$ with probability 0.3 because A is $\bar{a}$, and C is c .

The explanations that are extracted using EBI are independent of the inference algorithm. Furthermore, because the inference result and the assigned Markov values account for the same evidence, the generated EBI explanations are consistent with the inferences.

4.3 Explaining BN Compensations during Inferences

A significant advantage of EBI is that it can be readily extended to explain the mechanism by which a BN compensates for missing and erroneous inputs during inferences. EBI presents interested users with explanations on how the BN computes the distribution of variables whose values are not supplied. In addition, EBI highlights those situations in which the network detects apparent errors in one or more variables in the target variable's Markov blanket, and explains how the network compensates for them during inferences. In this section, we discuss how the EBI procedure explains these hidden BN processes.

Figure 4.6: Example for Explaining BN Compensations.

4.3.1 Explaining BN Compensation of Missing Inputs

An input is *missing* during inference if its value is not given as evidence. During inference, the node with the missing value is assigned a Markov value based on the interaction with its Markov nodes. EBI's treatment of missing inputs is to first explain the inference on the target variable as before using the target's assigned Markov values, and then provide an option for the users to retrieve further explanations for each missing Markov input.

Figure 4.6 shows a BN where the target is explained using the values of its Markov nodes 0, 1, 2 and 3. If node 0 is missing during an inference, its assigned value is explained using its own Markov nodes. EBI highlights missing Markov values as *weaknesses* in the evidence. Procedure 5 summarizes EBI's explanation of missing value compensations.

Procedure 5 Recursive-EBI (BN, V , n)

Input: BN, evidence V , and the node n to be explained.

Output: Explanation for the BN inference on n .

if n is not the target node **then**

Explain the BN's computed posterior distribution for n as a missing value compensation.

else

Explain n as the target node.

Highlight any missing Markov node m .

if the user requests an explanation on m **then**

Recursive-EBI (BN, V , m).

While the basic EBI procedure emphasizes explanation in terms of the target's Markov nodes, *Recursive-EBI* allows the explanation process to propagate through the BN. This provides an interactive mechanism to explain how specific missing values are compensated by the BN. Alternatively, one can employ a graph tracing algorithm to explain *all* the missing inputs, while still benefiting from EBI's exploitation of variable independencies.

4.3.2 Explaining BN Compensation of Erroneous Inputs

An input is potentially *erroneous* if its observed values carry a certain probability or rate of being incorrect. For specific applications like those involving sensor measurements, it is possible to specify a sensible threshold to filter away low natural background variations in readings, such that we are able to identify those inputs with a higher inconsistency as having a greater potential to be erroneous. BN handles a potentially erroneous input by entering its value as a *likelihood finding* instead of a *specific finding*. It is the likelihood ratio of the input states, and not the specific likelihood of each state, that is taken as evidence [65]. The likelihood ratio represents the relative probability of observing a value of an input given each of its possible states. This soft evidence allows the beliefs of that input to be affected by other nodes. Such exploitation of feature dependencies to correct inconsistent observations is related to the detection of inconsistencies in sensor data [143].

BN supports two types of error compensation. With reference to Figure 4.6, suppose that the value for node O is potentially in error and is entered as a *likelihood finding*. Its beliefs are open to “correction” by the other nodes, such that its value may be revised to become more consistent with the other findings. We term this as *Type I compensation*.

Definition (Type I Compensation): We say that Type I compensation of an erroneous input occurs, when its value is corrected to an assigned value after the inference.

Suppose that the value of node O is not corrected but the BN nevertheless produces the “right” inference. In other words, the effect of the error node O has been compensated by nodes I , 2 and 3 in deriving the target value. We term this as *Type II compensation*.

Definition (Type II Compensation): We say that Type II compensation of an erroneous input occurs, when its effect on the value of a target node has been overcome by other inputs to the target after the Bayesian network inference or recommendation.

EBI explains Type I and Type II compensations differently. To explain Type I compensation when an assigned Markov value differs from its finding, Recursive-EBI is enhanced to highlight the correction and explain the underlying compensation mechanism (see Procedure 6). The *Enhanced-EBI* procedure presents users with further explanations when there are corrections in the target’s Markov values due to Type I compensations.

CHAPTER 4. EXPLAINING RECOMMENDATIONS IN BAYESIAN NETWORKS

Procedure 6 Enhanced-EBI (BN, V , n)**Input:** BN, evidence V and the node n to be explained.**Output:** Explanation for the BN inference on n .

```

if  $n$  is not the target node then
  if  $n$  is missing from the evidence then
    Explain the assigned value for  $n$  as a missing value compensation.
  else if  $n$  has its value corrected then
    Explain the correction on  $n$  as an erroneous value compensation.
else
  Explain  $n$  as the target node.
for each corrected Markov node  $c$  do
  Enhanced-EBI (BN,  $V$ ,  $c$ ).
Highlight any missing Markov node  $m$ .
if user requests an explanation on  $m$  then
  Enhanced-EBI (BN,  $V$ ,  $m$ ).

```

For Type II compensations, it suffices for Enhanced-EBI to explain the target value as per Procedure 5. The reason is that EBI only considers the *assigned* Markov values after the beliefs in the BN have been updated based on the current evidence, regardless of whether they are likelihood findings, specific findings, or are inferred from other inputs.

4.3.3 Discussion

The simple strategies of EBI give rise to a number of apparent limitations, though most of the concerns can be addressed through straightforward extensions. They include:

- Explaining with most-probable values may be insufficient when there are more than one most-probable value, or when feature values are many and the highest posterior probabilities are too low for the explanation to be convincing. In the former case, presenting more than one possible explanation can be a viable solution. In the latter case, the main concern is whether the presented posterior of the target variable is too low to be convincing. However, considering that this low posterior has provided the very basis for the inference, presenting the low posterior can in fact alert the user to the low degree of confidence in the BN conclusion and aid the user's judgement.
- EBI explains the inferred target value and the underlying compensation processes using a local view of the target and its Markov nodes. Nevertheless, it is possible for certain

user to prefer (even an incomplete) global view of the inference process instead of a local view when working in a given context. A user-centric solution can entail the combination (a hybrid) of global and local views to the inference process.

- EBI is designed to effectively explain the inference on a single target node, but a user may require explanations for the value assignments in multiple targets instead. A straightforward solution would be to generate an independent EBI explanation for each of these target variables, by treating the rest of the variables as missing values in each round. However, EBI's recursive treatment of missing value explanation may then generate a large number of complex explanations. An alternative solution would be to take advantage of the recursion capability of the procedure, by presenting first the explanations for a small subset of the targets, then allowing the users to recursively obtain explanations for the remaining related target variables.

Keeping in view that these and other limitations may arise during user interactions, we proceed to empirically validate EBI's explanation efficacy in natural problem domains.

4.4 Experimental Validation

In Section 4.4.1, we present our experimental procedures and describe the problem domains. Section 4.4.2 begins by comparing the prediction accuracies of BN and J4.8 DT when inputs are missing. The purpose for the comparison is to provide motivation for improving explanation of BN inferences rather than predicting with DT, which are known to be reliable classifiers that also provide users with comprehensible rules. The remainder of Section 4.4.2 discusses examples from various domains to validate if EBI can effectively explain the missing inputs. Following a similar organization, Section 4.4.3 compares BN against the J4.8 DT and validates the EBI explanations when there are erroneous inputs. Section 4.4.4 summarizes the explanation efficacy of EBI from the various experiments.

4.4.1 Methods and Procedures

We adopt the CaMML program [98] to learn BN from data and the Netica-J API [101] to predict with BN. We present results on the mushrooms data set from UCI [144], the Syskill-

Figure 4.7: A BN Learned From the Mushrooms Data.

Figure 4.8: A BN Learned From the Web Page Data.

Webert web page ratings data set [114], and the water treatment network [145]. The data is preprocessed so that BN and DT are compared fairly using the same partitions of training and testing examples, on a P4 3GHz XP PC with 1GB of physical memory.

The mushrooms data set comprises of 8124 examples. For each mushroom example, the target variable *Edibility* predicts whether the given mushroom is *poisonous* or *edible* based on a set of observational input variables. Both BN and DT learn on inputs *Odor*, *Spore-print-color*, *Stalk-surface-below-ring*, *Stalk-color-above-ring*, *Habitat* and *Cap-color* because these are known to possess the highest classification power. As *Odor* is the most important input (with a classification accuracy of 98.52%), we compare BN and DT when *Odor* is missing or in error. Figure 4.7 shows a BN learned from the mushrooms data.

The Syskill-Webert web page ratings data set comprises the HTML sources of real web pages and their corresponding ratings (“medium/cold”, “hot”) by a human subject. Web pages on the topic of goats provide seventy examples for our experiments. The attributes are English words extracted from the web pages, excluding stopwords from the SMART list [115]. We conduct multiple trials by training on randomly-chosen examples and then testing on the rest. In each trial, we use the information gain criterion from the original study [116] to select the 32 most informative words. Figure 4.8 shows a BN learned from the web page data set using the

CHAPTER 4. EXPLAINING RECOMMENDATIONS IN BAYESIAN NETWORKS

Figure 4.9: The Water BN in Our Experiments.

minimal set learning procedure described in Chapter 3. Here, we do not use the error discovery procedure introduced in that chapter, so the new error compensation results that are presented in this chapter should be viewed independently.

The water network models the processes inside a waste water purification plant. As shown in Figure 4.9, eleven inputs from the 0^{th} and 15^{th} time slices are used to predict the target *CBODD_12_30*. A total of 1000 examples are generated using the Netica-J API. As DT uses only *CKNI_12_15* and *CBODD_12_15* in determining the value of target *CBODD_12_30*, we compare the BN and DT when these inputs are missing or erroneous.

Prediction performance is measured by *prediction accuracy*, defined as the percentage of correctly classified test examples (the mean absolute error (MAE) for a discrete class is one minus its prediction accuracy). For the mushrooms data, we perform five rounds of two-fold cross-validations to compare BN and DT. For the web page data, we perform ten trials of sixty training and ten test examples. For the water data, we compare BN and DT over ten test sets of 100 examples each. For BN, the missing values are left out during prediction, whilst for DT we substitute missing values with the most frequent values in the corresponding training sets. For the error compensation experiments, training sets have no error while test sets have error inputs with error rates ranging from 0% to 50%.

We need a measure of explanation quality to evaluate the EBI explanations. A suitable measure is given by Chajewska and Draper [136] as the ratio of absolute differences:

$$f(o) = 1 - \left| \frac{P(o|\delta_B) - P(o|\delta_X)}{P(o|\delta_B) + P(o|\delta_A)} \right| \quad (\text{Eq. 4.9})$$

where δ_A is the state of the BN before inference, δ_B is the state after inference, δ_X is the state in the explanation, o is the predicted target value, and $P(o|\delta)$ is the predicted value's probability given δ . Eq. 4.9 evaluates to zero if $P(o|\delta_X)$ equals $P(o|\delta_A)$ (the explanation is worthless as it

Table 4.1: Prediction Accuracy Under Missing Inputs. *Complete*: All Input Values are Present; *Missing*: Important Input Values are Missing.

	BN (%)		DT (%)	
	Complete	Missing	Complete	Missing
Mushrooms	99.45	95.76	99.90	54.39
Water	92.30	70.20	92.30	38.80
Web Page	75.00	71.15	72.69	66.92

Figure 4.10: BN After Reversal of the Arc Coming Out of *Edibility* in Figure 4.7.

Figure 4.11: BN After Reversal of Arcs Coming Out of *Odor* in Figure 4.7.

cannot explain the difference in $P(o|\delta_A)$ and $P(o|\delta_B)$); it peaks when $P(o|\delta_X)$ equals $P(o|\delta_B)$, is symmetrical about the peak, and falls linearly away from it.

4.4.2 Observations under Missing Inputs

The prediction accuracies of BN and DT for the scenarios with and without missing input in the data sets are summarized in Table 4.1. The results agree with observations in Chapter 3 that BN and DT perform similarly when given complete data for prediction, but BN's ability to capture feature dependencies and compensate for missing values using other available inputs enables it to outperform DT when important inputs are missing.

For the mushrooms data, BNs learned in the various validation cycles have a structure similar to Figure 4.7. Knowing that assigned *Odor* values can be explained by its Markov nodes, the structure does not tell us *how* these values explain the assigned value of *Odor*, nor *how* this in turn explains the *Edibility* inference. EBI brings out the missing insights.

Figures 4.10 and 4.11 show the networks after the BN in Figure 4.7 is restructured to explain the values of *Edibility* and *Odor*, respectively. Table 4.2 shows an EBI explanation when

CHAPTER 4. EXPLAINING RECOMMENDATIONS IN BAYESIAN NETWORKS

Table 4.2: Explanation Generated by EBI for Mushrooms. *Odor*'s Value is Missing.

BN Inference with EBI Explanation	Quality of Explanation
BN predicts Mushroom is poisonous with probability 1.000 because <u>Odor is foul</u> with probability 1.000, Stalk-surface-below-ring is silky with probability 1.000, Habitat is grasses with probability 1.000, and Stalk-color-above-ring is brown with probability 1.000.	1.000
Missing values compensated: BN infers Odor as foul with probability 1.000 because Spore-print-color is chocolate with probability 1.000, Cap-color is yellow with probability 1.000, and Stalk-surface-below-ring is silky with probability 1.000.	

Table 4.3: Explanation Generated by EBI for Water. *CKNI_12_15*'s Value is Missing.

BN Inference with EBI Explanation	Quality of Explanation
BN predicts CBODD_12_30 is 25mg/l with probability 0.549 because C_NI_12_15 is 6 with probability 1.000, CBODD_12_15 is 30mg/l with probability 1.000, CKNI_12_15 is 30mg/l with probability 0.600, and CBODN_12_15 is 5mg/l with probability 1.000.	1.000
Missing values compensated: BN infers CKNI_12_15 is 30mg/l with probability 0.600 because CKNI_12_00 is 30mg/l with probability 1.000, C_NI_12_15 is 6 with probability 1.000, and CBODD_12_15 is 30mg/l with probability 1.000.	

Odor is missing. For illustration, the line under *Odor* indicates that compensation has occurred. Only the explanation for *Edibility* is given in the initial feedback. If the user is interested in the reason for the assigned *Odor* value, EBI presents an explanation for the compensation as shown. Although we can present the posteriors for all the Markov nodes of the missing input, we have removed the target node from the EBI missing value explanation to avoid confusing the users. In the above example, *Edibility* is independent of *Spore-print-color* and *Odor* is independent of *Habitat* given the rest of their Markov values. The interactions among the Markov nodes allow EBI to simplify its explanation.

Table 4.3 shows an EBI explanation for the water BN's prediction on *CBODD_12_30* where *CKNI_12_15* is missing. As shown in Figure 4.9, *CBODD_12_30* and *CKNI_12_15* have five and six Markov nodes, respectively. EBI deduces that not all the Markov nodes are needed to

Table 4.4: Explanation Generated by EBI for Web Page. *development*'s Value is Missing.

BN Inference with EBI Explanation	Quality of Explanation
BN predicts rating is cold/medium with probability 0.814 because dairy is absent with probability 1.000, <u>development is absent</u> with probability 0.936, and farm is absent with probability 1.000.	1.000
Missing values compensated: BN infers development is absent with probability 0.936 because management is absent with probability 1.000, dairy is absent with probability 1.000, and farm is absent with probability 1.000.	

explain the two values and presents its simplified explanations as shown. The missing status of *CKNI_12_15* is highlighted in the explanation for *CBODD_12_30*, and only interested users are presented with the missing value explanation in the table.

Table 4.4 shows an EBI explanation for the web page data. As shown in Figure 4.8, *rating* has *dairy*, *development* and *farm* as its Markov nodes. The explanation in Table 4.4 is for a certain case where *development* is missing from the evidence. Likewise, the missing Markov value is highlighted and further explanation is presented only upon user request.

As shown, EBI explains missing inputs via a flexible and user-centered mechanism, which helps users to better appreciate how missing values are compensated by the BN.

4.4.3 Observations under Erroneous Inputs

Figure 4.12 summarizes the results for the domains under the effect of erroneous inputs. BN outperforms the DT under identical error conditions because it captures and exploits the dependencies among its variables. Figures 4.12.a and 4.12.b show that BN maintains its accuracy despite the input errors, whilst DT suffers as errors intensify. BN's robustness is most pronounced for mushrooms and less so for web pages, possibly due to the weaker dependency among words. Likewise, Figure 4.12.c suggests that the dependencies among water treatment processes are weak but BN outperforms DT when the error rate is high.

Table 4.5 presents an example from the mushrooms experiments where an error in *Odor* leads to a different inference on *Edibility*. The inference based on the correct *Odor* value of

CHAPTER 4. EXPLAINING RECOMMENDATIONS IN BAYESIAN NETWORKS

4.12.a: For Mushrooms.

4.12.b: For Web Page.

4.12.c: For Water.

Figure 4.12: Prediction Accuracy With Erroneous Inputs in the Test Data.

foul is that the mushroom is *poisonous* (Table 4.5(a)). However, when *Odor* is wrongly entered as *none*, BN predicts that the same mushroom is *edible* (Table 4.5(b)). In this case, when the error in *Odor* is entered as a specific finding, EBI explains the *Edibility* value as follows:

BN predicts Mushroom is edible with probability 0.965 because

Odor is none with probability 1.000,

Spore-print-color is chocolate with probability 1.000,

Stalk-surface-below-ring is silky with probability 1.000,

Habitat is woods with probability 1.000, and

Stalk-color-above-ring is brown with probability 1.000.

If we instead enter the erroneous value of *none* for *Odor* as a likelihood finding, *Odor* is corrected to *foul* during the inference and we get the same prediction as with no error (Table 4.5(c)). EBI highlights this compensation as shown in the first explanation of Table 4.7, where the value of *Habitat* is found to be unnecessary for explaining the correction in *Odor*.

CHAPTER 4. EXPLAINING RECOMMENDATIONS IN BAYESIAN NETWORKS

Table 4.5: An Example From Mushrooms Data where a Type II Compensation Gives a Different Prediction When *Odor* is In Error.

(a) No Error Among the Input Values.		
<i>Odor</i>	<i>Spore-print-color</i>	<i>Edibility</i>
foul	chocolate	poisonous
(b) Wrong <i>Odor</i> - As Specific Finding.		
<i>Odor</i>	<i>Spore-print-color</i>	<i>Edibility</i>
none	chocolate	edible
(c) Wrong <i>Odor</i> - As Likelihood Finding.		
<i>Odor</i>	<i>Spore-print-color</i>	<i>Edibility</i>
foul	chocolate	poisonous

Table 4.6: An Example From Mushrooms Data Where a Type II Compensation Gives an Identical Prediction When *Odor* is In Error.

(a) No Error Among the Input Values.		
<i>Odor</i>	<i>Spore-print-color</i>	<i>Edibility</i>
none	black	edible
(b) Wrong <i>Odor</i> - As Specific Finding.		
<i>Odor</i>	<i>Spore-print-color</i>	<i>Edibility</i>
fishy	black	edible
(c) Wrong <i>Odor</i> - As Likelihood Finding.		
<i>Odor</i>	<i>Spore-print-color</i>	<i>Edibility</i>
none	black	edible

Corresponding to the second explanation in Table 4.7, Table 4.6 shows another example from the mushrooms experiments. Just as when *Odor* is *none* (Table 4.6(a)), the BN predicts that the mushroom is *edible* when the wrong *Odor* value of *fishy* is entered as a specific finding (Table 4.6(b)). Type II compensation, which involves inputs to the target interacting to produce the right inference, helps the BN in this case to maintain the same prediction even when *Odor* is in error. EBI explains this inference as follows:

BN predicts Mushroom is edible with probability 0.523 because

Odor is fishy with probability 1.000,

Spore-print-color is black with probability 1.000,

Stalk-surface-below-ring is fibrous with probability 1.000,

Habitat is grasses with probability 1.000, and

Stalk-color-above-ring is white with probability 1.000.

When the wrong *Odor* value of *fishy* is entered as a likelihood finding, the BN corrects the *Odor* value to *none*, in addition to making the correct inference that the mushroom is edible (Table 4.6(c)). EBI highlights this compensation as shown in the second explanation of Table 4.7. In the above example, EBI simplifies its explanation for *Edibility* by exploiting context-specific independencies among the nodes in *Edibility*'s Markov blanket (the parents of *Edibility* in Figure 4.10) to omit *Stalk-color-above-ring*. Similarly, EBI deduces from the

Table 4.7: Explanations Generated by EBI for Mushrooms. *Odor*'s Value is In Error.

BN Inference with EBI Explanation	Quality of Explanation
BN predicts Mushroom is poisonous with probability 1.000 because <u>Odor is foul</u> with probability 1.000, Spore-print-color is chocolate with probability 1.000, Stalk-surface-below-ring is silky with probability 1.000, Habitat is woods with probability 1.000, and Stalk-color-above-ring is brown with probability 1.000.	1.000
Erroneous values compensated: BN corrects <i>Odor</i> from none to foul because given Spore-print-color is chocolate, Cap-color is yellow, and Stalk-surface-below-ring is silky, <i>Odor</i> is none with probability 0.000, and <i>Odor</i> is foul with probability 1.000.	
BN predicts Mushroom is edible with probability 0.999 because <u>Odor is none</u> with probability 0.999, Spore-print-color is black with probability 1.000, Stalk-surface-below-ring is fibrous with probability 1.000, and Habitat is grasses with probability 1.000.	1.000
Erroneous values compensated: BN corrects <i>Odor</i> from fishy to none because given Spore-print-color is black, Cap-color is white, and Stalk-surface-below-ring is fibrous, <i>Odor</i> is fishy with probability 0.000, and <i>Odor</i> is none with probability 0.999.	

discovered interactions amongst *Odor*'s Markov nodes (the parents of *Odor* in Figure 4.11) that *Habitat* is not necessary for explaining the correction in *Odor*.

Table 4.8 shows an EBI explanation for water. As shown in Figure 4.9, *CBODD_12_30* has five Markov nodes while the erroneous *CBODD_12_15* has ten. EBI exploits CSI to omit *CNOD_12_15* when explaining *CBODD_12_30*, and it selects just three out of the ten Markov nodes of *CBODD_12_15* for explaining its correction. Table 4.9 presents an EBI explanation for the web page problem. Likewise, EBI not only explains the inferred *rating*, it also highlights and explains the correction in *farm*'s value during the inference.

4.4.4 Explanation Efficacy

We summarize the empirical statistics from the experiments. Table 4.10 shows the average time taken to generate an EBI explanation, and Tables 4.11 and 4.12 present the savings that

CHAPTER 4. EXPLAINING RECOMMENDATIONS IN BAYESIAN NETWORKS

Table 4.8: Explanation Generated by EBI for Water. *CBODD_12_15*'s Value is In Error.

BN Inference with EBI Explanation	Quality of Explanation
BN predicts <i>CBODD_12_30</i> is 25mg/l with probability 0.878 because <i>C_NI_12_15</i> is 6 with probability 1.000, <i>CBODD_12_15</i> is 25mg/l with probability 0.984, <i>CKNI_12_15</i> is 20mg/l with probability 1.000, and <i>CBODN_12_15</i> is 20mg/l with probability 1.000.	1.000
Erroneous values compensated: BN corrects <i>CBODD_12_15</i> from 30mg/l to 25mg/l because given <i>CKNI_12_00</i> is 20mg/l, <i>CBODD_12_00</i> is 25mg/l, and <i>CKNI_12_15</i> is 20mg/l, <i>CBODD_12_15</i> is 30mg/l with probability 0.016, and <i>CBODD_12_15</i> is 25mg/l with probability 0.984.	

Table 4.9: Explanation Generated by EBI for Web Page. *farm*'s Value is In Error.

BN Inference with EBI Explanation	Quality of Explanation
BN predicts rating is cold/medium with probability 0.984 because dairy is absent with probability 1.000, and <u>farm is absent</u> with probability 0.997.	1.000
Erroneous values compensated: BN corrects <i>farm</i> from present to absent because given pigs is absent, kids is absent, animal is absent, and dairy is absent, farm is present with probability 0.003, and farm is absent with probability 0.997.	

are due to CSI. We quantify the savings by measuring the explanation complexity (defined as the number of rule antecedents as per [146]) before and after exploiting CSI. Table 4.11 shows the percentage of cases with reduced explanation complexity after exploiting CSI, and Table 4.12 presents the reduction in complexity across all the cases.

The EBI explanations have an average quality of 100% (see formula (Eq. 4.9)). Table 4.10 shows that EBI generates its explanations in a matter of seconds, and Table 4.11 shows that as many as half of all test cases have benefitted from EBI's exploitation of CSI to simplify explanations. Table 4.12 shows that on average, EBI explanations enjoy substantial reduction in complexity by explaining only with significant Markov nodes. The results show that EBI produces concise rules to explain BN inferences in real time. Moreover, EBI is unique in its ability to explain missing and error input compensations.

Table 4.10: Average Time (in Units of Seconds) Taken for Generating Each EBI Explanation.

Mushrooms	1.397
Water	2.657
Web Page	0.001
Average	1.352

Table 4.11: Percentage of Cases With a Reduction in Explanation Complexity After Exploiting CSI.

Mushrooms	62.98
Water	71.14
Web Page	10.00
Average	48.04

Table 4.12: Explanation Complexity. *Before CSI*: Average Number of Antecedents Before Exploiting CSI; *After CSI*: Average Number of Antecedents After Exploiting CSI.

	Before CSI	After CSI	% reduction
Mushrooms	6.393	5.615	12.17
Water	7.355	5.629	23.47
Web Page	3.471	3.306	4.757
Average	5.740	4.850	13.47

4.5 Chapter Summary

While Bayesian network (BN) can achieve accurate predictions even with erroneous or incomplete evidence, explaining the inferences remains a challenge. Existing approaches fall short because they do not exploit variable interactions and cannot account for compensations during inferences. In this chapter, we present a novel Explaining BN Inferences (EBI) procedure for explaining how variables in the BN interact to reach conclusions.

The EBI procedure explains the inferred target value in terms of the influential nodes within its Markov blanket. EBI simplifies its explanations by exploiting context-specific independence relations that apply when there are specific value assignments to certain Markov nodes. Working back from the target, EBI presents the derivation of each intermediate variable in a layered set of explanations, which shows how each of the missing or erroneous values is compensated during the BN inference to arrive at the final recommendation.

We validate the EBI procedure on problems including mushroom classification, water purification and web page recommendation. The results show that EBI generates high quality, concise and comprehensible explanations for BN inferences within seconds. Furthermore, the EBI explanations effectively account for the underlying noise compensation mechanism that enables the BN to outperform alternative prediction systems, such as decision trees.

Chapter 5

An Effective Knowledge Discovery Process for Noisy Bio-Data Classifications

5.1 Introduction

As discussed in Section 2.3.4 of Chapter 2, *bio-data classifications*, or classification of biological processes based on genotypic features, has wide applications, e.g. infection detection, tumor identification. However, rampant noise in bio-data (e.g. due to chemical contamination, hybridization, and microarray spots irregularities) can lower the classification accuracy significantly [147]. In many cases, dependencies can be observed between genotypic features (the contexts of the classification task), and this input redundancy can cancel out some of the noise. While classifiers like decision tree and support vector machine [148] capture only the associations from features to class, Bayesian network [17] is able to capture the feature dependencies.

In this chapter, we introduce a Bayesian network (BN)-based knowledge discovery process for noisy biological data mining, consisting of two procedures from the preceding chapters that deal explicitly with noisy data. The first procedure is the one we have proposed for automatic noise correction, where a BN is trained to capture the dependencies among selected features in the data. Using the learned BN to predict each feature in turn, the procedure estimates each feature's error probability from the deviation between its recorded and predicted values in the training data, so that the appropriate uncertainty can be associated with subsequent findings.

The second key component in the proposed knowledge discovery process is our Explaining BN Inferences (EBI) procedure, which explains each BN recommendation based on the feature values within the target node's Markov blanket. Where the value of a feature differs from what

CHAPTER 5. AN EFFECTIVE KNOWLEDGE DISCOVERY PROCESS FOR NOISY BIO-DATA CLASSIFICATIONS

is stated during classification, the EBI procedure explains the value correction in a recursive manner. The EBI procedure's ability to explain both the BN classifications and the underlying noise correction mechanism is not found in other existing explanation methods, e.g. [139].

With the proposed knowledge discovery process, we seek to answer key concerns about the overall process's efficacy through empirical validation in the later part of this chapter:

- The noise discovery method fits a BN to the training data and claims that a feature is more likely to be erroneous if remaining features do not predict its values well. Can the approach correctly find all those features with noisy values? To answer this, we inject different severity of noise into the HIV-1 data from the Stanford HIV Drug Resistance Database [149] (<http://hivdb.stanford.edu/pages/asi/releaseNotes/>). We can then verify if the noise levels are correctly detected, and how BN performs under noisy conditions.
- The EBI explanation method relies on the target's Markov blanket to describe biological interactions. Do nodes that are captured in the target's Markov blanket have experimentally demonstrated associations with the target node? To answer this question, we verify whether BN learned on the HIV-1 data contain known resistance mutations as the target's Markov nodes (e.g., HIV-1 resistance to NRTI drugs is attributable to mutations in TAMs (e.g. M41L, D67N) and Q151M complex [150]). By learning BNs from many independent data partitions and summarizing the frequencies with which known mutations appear, we verify whether the target's Markov blanket consists of these salient mutations.
- Protein mass spectroscopy and DNA microarray represent two prevalent sources of bio-data. How does the proposed KDD process perform on the real-world mass spectroscopy and microarray data sets? To answer this, we validate the proposed process using real-world protein mass spectroscopy data for ovarian cancer detection, and microarray data for acute leukemia subtype classification. Protein analysis (proteomics) techniques depend on mass spectral data streams that are infested with substantial electronic and chemical noise due to contaminants and matrix effects [151, 152]. Likewise, existing gene expression data from high-throughput methods like DNA microarrays are extremely noisy due to microarray spot irregularities and other sources of experimental errors, e.g. arising

from processes including RNA-hybridization [153–155]. Our experiments here would validate if the proposed process works well for such noisy bio-data classification problems, although the same process can be effectively applied to any other noisy domains.

The rest of this chapter is organized as follows. First, Section 5.2 describes the existing solutions to the investigated problems of HIV-1 drug resistance prediction, ovarian cancer detection, and acute leukemia subtype classification. Section 5.3 presents our proposed knowledge discovery process, comprising the key steps of feature selection, learning feature dependencies in a BN, estimating the features’ error rates, predicting with the noisy features, and explaining the classification. Finally, Section 5.4 answers all the above mentioned concerns in turn.

5.2 The Problem Domains

5.2.1 HIV-1 Drug Resistance Prediction

A variety of computational methods have been used to predict a patient’s drug resistance based on mutations, including linear regression [156] and support vector machines [150]. Saigo et al. [150] commented that none of the existing methods “can achieve high accuracy and good interpretability at the same time”, and so they proposed *itemset-boosting* to extract linear regression rules with mutation associations (complex features) as predicates. Itemset-boosting assumes that predicates (complex/singular features) combine linearly to predict the outcome. There is no knowledge of how features *interact* to affect drug resistance, nor how the singular features in each mutation association are related.

To understand the role of drug-resistance mutations, other works have turned to the Bayesian networks [157–159]. Our works are significantly different in a number of ways. First, Deforche et al. [157–159] aim to understand the *evolution* of drug resistance during treatments. In contrast, we aim to discover a model of salient mutations that predicts a virus’s resistance to a specific drug, or a drug class. The mutations need *not* be *in vivo* reactions to previous consumption of the drug; in the data used, mutations are the results of site-directed mutagenesis, and the resistance levels are determined *in vitro* [156].

In addition, while Deforche et al. analyzed the protease inhibitors (e.g. nelfinavir), we are interested in the nucleoside reverse transcriptase inhibitor (NRTI) Tenofovir (TDF). Our

study on TDF drug resistance is interesting because it is known that for NRTIs, only the combinations of multiple mutations can increase HIV-1's resistance to drugs, and it is found that salient large mutation associations are more common for NRTIs than other drug classes [150]. While the prior BN works focus on learning and interpretation, we are the first to analyze BN classification performance for the prediction of HIV-1 resistance.

5.2.2 Ovarian Cancer Detection

We validate the KDD process based on the mass spectra data from the US National Cancer Institute's *Clinical Proteomics Program Databank* (<http://home.ccr.cancer.gov/ncifdaproteomics>) (Ovarian Data Set 8-7-02). The experiment aims to discover mass-to-charge (M/Z) patterns in blood serum to distinguish patients with ovarian cancer from those without. This is especially critical for early detection in women who are at high risk of ovarian cancer due to a family or personal history of cancer, and for women with genetic predisposition to ovarian cancer due to abnormal *BRCA1* and *BRCA2* genes [160].

Lilien et al. [161] develop a Q5 algorithm that uses principal components analysis [117] to reduce the dimensionality of the serum spectra before classifying using linear discriminant analysis [162]. The method's prediction confidence is defined in a normal Gaussian distribution centered at each class mean, where a higher threshold allows for more confidence at the expense of fewer classifiable samples. For the same ovarian data set, they report that a certain threshold exists for which Q5 classifies perfectly, but it is not clear how this crucial threshold could be determined in general. More importantly, principal components analysis generates new features that cannot be related back to the proteins.

Sorace and Zhan [163] use the Wilcoxon nonparametric test to find M/Z values having the largest differences between cancer and normal sera, and stepwise discriminant analysis to develop rules for diagnosis. For this data, they present two rules that classify perfectly. However, their rules involve proteins with M/Z values as low as only 2.792 and 2.823. As the authors and later Baggerly et al. [164] have highlighted, this can be a major concern because it is very difficult to offer any biological explanation for the observed difference in such a low M/Z region. As their method does not learn the protein interactions, it cannot be used to examine these features' relations to other proteins and ovarian cancer.

Li and Wong [165] compare the prediction performance of two rule-induction algorithms, the C4.5 [123] and their own PCL classifier, using this same ovarian cancer data. They report that the decision tree that is learned by C4.5 misclassifies ten out of about 25 test samples, while their PCL classifier, which uses multiple significant rules as a committee, misclassifies only four samples. Their induced rules are also more interpretable compared to the functions that are defined within kernel-based methods. However, like the C4.5, their rule-based classifier model is susceptible to the noises in the protein values during diagnoses, and it cannot encode the causal interactions among important proteins.

5.2.3 Acute Leukemia Subtype Classification

More than 40,000 new leukemia cases and 20,000 leukemia-related deaths were expected in USA in the past year [166]. The acute leukemia, or the cancer of the blood, possesses heterogeneous subtypes with diverse therapy responses [167]. Misdiagnosis can result in suboptimal therapies that lead to excess toxicity and low survival rate [168], so recognizing the right subtype is critical. To date, studies on the classification of acute leukemia into its main subtypes based on gene expressions have been highly promising [168–170].

We analyze two sets of acute leukemia microarray gene expression data: the ALL/AML data set [168] from the Broad Institute Cancer Program (<http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi>), and the Pediatric ALL (childhood leukemia) data set [169] from St. Jude Children Research Hospital (<http://www.stjude-research.org/data/ALL1>). Various learning methods have been applied to these data, e.g. support vector machines [169]. To the best of our knowledge, this is the first time BN is applied to analyze the data sets.

Extremely impressive accuracies have been published for both data sets, so these are not unusually difficult problems. Nevertheless, using our integrated knowledge discovery process, BN yields a predictive accuracy comparable or superior to all past results. Most importantly, in contrast to the other methods like SVM and DT, which do not encode the dependencies among genes and thus cannot explain their predictions based on the genes' interactions, probabilistic rules can be automatically generated to explain BN inferences, in complement to the graphic presentation of gene interactions in the Bayesian networks.

Figure 5.1: Knowledge Discovery from Noisy Data.

5.3 The BN-Based Knowledge Discovery Process

Procedure 7 Knowledge Discovery from Noisy Data

Input: Training data ($train$), test data ($test$), error threshold (t).

Output: BN learned from $train$, erroneous features (F) with error rates (R), inferences (I) on $test$, and explanations (E) for I .

Step 1: Select Useful Features

Let A be the set of attributes in $train$.

for each attribute $a \in A$ **do**

if a does not separate the target classes **then** $A = A - \{a\}$.

Step 2: Capture Feature Dependencies

Select the top- k most-informative attributes (top_A).

Learn BN from $train$ based on top_A .

Step 3: Estimate Error Rates of Features

Identify F and estimate their R values using Procedure 8.

Step 4: Predict with the Noisy Features

for each test case $c_i \in test$ **do**

 Estimate the likelihoods of F , based on R and case c_i .

 Enter c_i into BN to obtain inference i ; $I = I + \{i\}$.

 Generate explanation e for inference i ; $E = E + \{e\}$.

return BN and the sets F , R , I , and E .

Figure 5.1 illustrates our knowledge discovery process. As summarized in Procedure 7, the proposed KDD process systematically brings the noisy data through the four steps of data preprocessing, model extraction, noise discovery, and diagnosis with explanation.

5.3.1 Step 1 - Select Useful Features

A thorough knowledge discovery approach has to examine all candidate features so as not to overlook the handful of discriminatory features, but feature selection is necessary as current BN structure learning algorithms are unable to handle large number of nodes, and discretization is needed as existing learning algorithms mostly assume discrete variables.

We implement the entropy-based discretization and feature selection method [171] in this work, as it is known to be effective for: i) ignoring trivial variations in intensity due to noise,

and ii) sifting through high-dimensional data to select important features that can distinguish samples from different classes. The method, which has been successfully used to preprocess other bio-data [170, 172], combines the entropy-based splitting criterion of the C4.5 decision tree [123] with the minimum description length (MDL) stopping criterion. It finds the optimal cutting point for every attribute so as to maximize its separation of the classes. Features without cutting points are discarded, effectively reducing the dimensions while converting any continuous attribute values into discrete variables. Other effective feature selection methods can also be employed for this step, e.g. [173].

5.3.2 Step 2 - Capture Feature Dependencies

The dependencies amongst features can be captured qualitatively in the directed acyclic graphical structure of a BN, and quantitatively by the conditional probabilities associated with each variable or *node*. The BN describes local feature interactions well, where each feature directly depends on a small number of neighboring nodes. In addition, the BN's probabilistic reasoning enables it to handle feature value uncertainties. In this work, we adopt the CaMML BN discovery program [65, 98] for supervised BN learning from data.

CaMML stochastically searches over the entire space of causal models to look for the best model that maximizes a Minimum Message Length (MML) posterior metric [65]. For each real model it visits, CaMML computes a representative model and counts only on these representatives to overlook the trivial variations. The MML posterior of each representative is computed as the total MML posterior of its members. This total posterior approximates the probability that the true model lies within the MML equivalence class of a representative, so the representative with the highest posterior is the best model. Alternative methods like constraint-based BN learning can also be adopted in this step.

5.3.3 Step 3 - Estimate Error Rates of Features

A feature in a given data set is potentially erroneous or noisy if it carries observations with some error probability. An error rate or probability of e for a feature f implies that observations for f are wrong $e*100\%$ of the time. In practice, we do not know how many and which features

are noisy, so we need to be able to discover multiple error features, and to estimate each of their error rates. As shown in Procedure 8, the error discovery procedure takes in the training examples and the corresponding learned BN. Predicting with the BN and using the proportion of misclassified training examples to estimate each feature's error rate, we identify the feature that is most likely to be erroneous, based on the intuition that the noisiest feature would be least consistent with the joint distribution.

For each training example, we enter the evidence minus the feature in question and exploit the captured causal dependencies between that feature and the rest of the network to check for inconsistency. After the *top* most erroneous feature is identified, the procedure accounts for the uncertainty in that feature's value by entering its value for each training example as a *likelihood finding*, or a *soft evidence*, while it searches for the *next* most likely error feature in a similar manner. This process is repeated to find the third most erroneous feature, and so on. It terminates when the error rates for all present features have been estimated, or until the minimum detected error rate falls below a given threshold value. The error threshold thus allows experts to filter or ignore trivial degrees of natural inconsistencies in the features' values; in our experience, a suitable threshold would be 0.1 or less, so that as many of the erroneous features are identified as possible.

5.3.4 Step 4 - Predict with the Noisy Features

5.3.4.1 Likelihoods Estimation

We expect any unseen or test examples to be noisy as well, with noise characteristic similar to the training data. Inputting features' values as *specific findings*, or *hard evidence*, is very likely to result in the wrong predictions if those findings are incorrect. A BN allows us to specify potentially noisy findings as *likelihoods* so as to reflect their uncertainties.

Suppose a feature f is observed to possess one of its three possible values $V: \{0, 1, 2\}$ (this observation is hereafter denoted as o). The *likelihood* of each feature value $v_f \in V$ is defined as the probability of o given that v_f is f 's true value. The likelihood values of the observation o are hence computed as $\{\text{prob}(o|f = 0), \text{prob}(o|f = 1), \text{prob}(o|f = 2)\}$.

Suppose the observation is that $\{f = 1\}$, and that feature f carries an estimated error rate of e (the rate e can be found using Procedure 8). The likelihoods for $o: \{f = 1\}$ are then

Procedure 8 Automatic Error Discovery Procedure

Input: Training data ($train$), BN from $train$, error threshold (t).

Output: Erroneous features (F), and their error rates (R).

Step 1: Identify the top erroneous feature f_{top}

Set the first feature, f_1 , as f_{top} .

for each record $\in train$ **do**

 Cover-up f_1 and predict its value using BN .

$P_{err}(f_1)$ = fraction of $train$ that f_1 is misclassified.

$max_err = P_{err}(f_1)$.

for each remaining feature f_i **do**

for each record $\in train$ **do**

 Cover-up f_i and predict its value using BN .

$P_{err}(f_i)$ = fraction of $train$ that f_i is misclassified.

if $P_{err}(f_i) > max_err$ **then**

$f_{top} = f_i$, $max_err = P_{err}(f_i)$.

if $max_err < t$ **then return** F and R as empty sets.

else $F = F + \{f_{top}\}$, $R = R + \{max_err\}$.

Step 2: Identify the remainder of sets F and R

$min_err = P_{err}(f_{top})$.

while $\exists$ feature $\notin F \wedge min_err \geq t$ **do**

for each feature f_i **do**

for each record $\in train$ **do**

 Estimate likelihoods L for feature values within F .

 Cover-up f_i and predict its value using BN and L .

$P_{err}(f_i)$ = fraction of $train$ that f_i is misclassified.

 Set first feature $\notin F$, f_1 , as f_{next} ; $max_err = P_{err}(f_1)$.

for each remaining feature $f_i \notin F$ **do**

if $P_{err}(f_i) > max_err$ **then**

$f_{next} = f_i$, $max_err = P_{err}(f_i)$.

$F = F + \{f_{next}\}$; $R = \emptyset$.

for each feature $f_i \in F$ **do**

$R = R + \{P_{err}(f_i)\}$.

 Set $min_err = P_{err}(\text{first feature} \in F)$.

for each remaining feature $f_i \in F$ **do**

if $P_{err}(f_i) < min_err$ **then**

$min_err = P_{err}(f_i)$.

Set first feature $\in F$, f_1 , as f_{least} ; $min_err = P_{err}(f_1)$.

for each remaining feature $f_i \in F$ **do**

if $P_{err}(f_i) < min_err$ **then**

$f_{least} = f_i$, $min_err = P_{err}(f_i)$.

if $min_err < t$ **then** $F = F - \{f_{least}\}$, $R = R - \{P_{err}(f_{least})\}$.

return the non-empty sets F and R .

computed as $\{P(0) * e * P(1), P(1) * (1.0 - e), P(2) * e * P(1)\}$, where $P(v_f)$ is the prior probability of v_f (approximated by the proportion of training data for which v_f is observed). This is because the probability of observing $\{f = 1\}$ when value of f is actually 0 or 2 is the probability that one of these other values is present in the example but we wrongly record f as 1. We can generalize this likelihoods estimation as follows:

Likelihood of the observed $v = P(v) * (1.0 - e)$, and

Likelihood of value $v' \in \{V - v\} = P(v') * e * P(v)$.

Entering a zero error rate is akin to entering the finding with complete certainty. By properly accounting for the confidence in feature values, our automatic error discovery and likelihoods estimation procedure (hereafter referred to as our Bayesian network-based noise correction (BN-NC) procedure) enables the BN to predict accurately on noisy data.

5.3.4.2 Explaining the BN Inferences

We integrate the novel Explaining BN Inferences (EBI) procedure from Chapter 4 to auto-generate explanations for the BN inferences. EBI explains the value of a target node in terms of just the contextually influential nodes in its Markov blanket (the target's parents, children, and spouses). This is necessary and sufficient because conditional independence implies that the Markov values form a minimal set that fully explains the inferences. To simplify its explanations, EBI exploits context-specific independence as reflected in the target's conditional probabilities. Working back from the target variable, EBI shows the derivation of each intermediate variable in terms of their own respective Markov nodes, thereby explaining how missing and noisy evidence values are corrected during inferences.

The EBI procedure consists of three key steps. First, the procedure restructures the local dependencies around the target via arc reversals, in such a way that all its Markov nodes become its parents while maintaining the joint distribution. The resulting conditional probability table (CPT) of the target describes its probability distribution over all its Markov value combinations. Next, EBI reduces the complexity of the generated rules by learning from the target CPT a decision tree (DT) [123] that captures the context-specific independence among Markov nodes for each target value. Finally, for each BN inference or recommendation, EBI compares

Figure 5.2: Test Prediction Accuracies for TDF.

the assigned Markov values against the DT for the inferred target value, and explains the recommendation in terms of the significant Markov values corresponding to those nodes on the path describing the current context.

5.4 Experimental Validation

5.4.1 HIV-1 Drug Resistance Prediction

In our experiments, resistance to the nucleoside reverse transcriptase inhibitor, Tenofovir (TDF), is predicted based on the observed mutations in isolates' DNA sequences. Our data set is the same as the one used previously by Rhee et al. [156] and Saigo et al. [150].

Within identical setups of noisy environment, we compare the cross validation classification accuracy of Bayesian network (BN) versus BN with noise correction (BN-NC), using J4.8 decision tree [123] as a baseline. Performance from ten independent rounds of five-fold cross validation are computed for statistical stability. In each round, the data set is randomly partitioned into five equal-sized parts or *folds*. We use one part to test and the rest for training, and repeat this for each part in turn to get the average accuracy.

Figure 5.2 presents the results obtained by applying the different methods on identical training and test splits with various degrees of introduced noise. The proposed correction technique consistently enables BN-NC to substantially outperform both the J4.8 DT and the BN with no noise correction. In the figure, BN with noise correction maintains an accuracy of close to 0.7 when the error rate is 0.3 or 0.4, whilst both DT and the BN with no noise correction suffer a much greater drop in accuracy as more errors are introduced.

Figure 5.3: Error Rate Increments Detected for TDF.

Figure 5.4: The Initial Error Rates Detected for TDF.

To verify whether the proposed error discovery procedure can indeed sense the introduced noise correctly, we compute the difference between the detected error rates for various mutations at each degree of noise and those when no noise is added. If the noise detection method works properly, then we would expect a steady rising trend in each mutation’s detected error rate as we introduce more noise. As shown in Figure 5.3, this is indeed the case. For each of the seven mutations in which noise has been added, the detected error rate increases steadily as more noise are introduced into the data. The method also correctly concludes that no additional noise has been added at all to mutation T215Y, as reflected by the only flat line in the figure. The differences in slopes between the mutations are due to differences in the initially detected error rate, i.e., when no additional noise is introduced. This is because the noise in the data are introduced by randomly flipping the mutation values in the samples according to the intended rate, which means that some of the initial error values are actually flipped in the process. A

CHAPTER 5. AN EFFECTIVE KNOWLEDGE DISCOVERY PROCESS FOR NOISY BIO-DATA CLASSIFICATIONS

mutation with a smaller initial rate would have a steeper slope in Figure 5.3, since most of the randomly administered flips would add on to the existing errors. We arrange the seven mutations in increasing order according to their gradients in Figure 5.3, and plot their respective initial error rates in Figure 5.4. Indeed, the lower the initial rate for a mutation, the steeper the detected error increment would be. The results verify that our error discovery procedure (Procedure 8) accurately detects the noise that are introduced into the values of the mutations, within limits of the data's inherent noise characteristics.

In drug resistance studies, a mutation that is selected by the virus as the first mutation to increase resistance against a drug is referred to as a *major mutation*, and a mutation that is selected only in the presence of some other mutations is a *minor mutation*. We now investigate whether known major and minor mutations are correctly recovered as part of the target node's Markov blankets in the learned BN models; this would in turn vouch for the rationale behind the proposed Markov blanket-based EBI explanation approach.

Biologically, HIV-1 excises or neutralizes NRTIs via two mechanisms i) the thymidine-associated mutations (TAMs), comprising D67N, M41L, L210W, T215Y/F, K70R and K219Q, and ii) Q151M complex, comprising Q151M, A62V, V75I, F77L, and F116Y [150]. It must thus be established that the discovered Markov blankets contain these mutations.

We analyze the fifty independently learned BNs from our cross-validation experiments, and tally the number of times that each mutation appears in the learned Markov blankets. Table 5.1 summarizes the findings. As can be seen from this table, the two major singular mutations are the D67N and M41L, which are both members of the TAMs. Combinations of TAMs or T69i and TAMs are observed in all the Markov blankets, whilst no Q151M complex mutation can be found, suggesting that for TDF, the TAMs are the predominant resistance pathways selected by HIV-1. Interestingly, mutation L210W always appears in tandem with D67N, the most common mutation in the Markov blankets. Likewise, T69i (insertion of dipeptide between positions 69 and 70) always appears together with TAMs, which qualifies it as a minor mutation. Indeed, biological experiments have shown that the combination of T69i and TAMs provides strong resistance to all the NRTI drugs [174].

Table 5.1: Frequency of Mutations in the Markov Blankets.

Mutation	Frequency	TAMs?	Q151M-Complex?
D67N	40	Yes	No
M41L	30	Yes	No
L210W	26	Yes	No
T69i	5	Yes	No
L210W & D67N	26	Yes	No

Table 5.2: The Top 10 Proteins Selected in Decreasing Order of Information Gain.

Protein (<i>M/Z</i>)	Information Gain	Cutting Points
MZ261.88643	0.8190	0.3771, 0.4011, 0.4388
MZ262.18857	0.7627	0.4317, 0.5107
MZ245.24466	0.7381	0.4508
MZ244.95245	0.7381	0.4350
MZ245.8296	0.7219	0.3908
MZ245.53704	0.7219	0.5052
MZ244.66041	0.7202	0.2098, 0.3608
MZ246.12233	0.6903	0.3478
MZ435.07512	0.6693	0.3647
MZ434.29682	0.6650	0.4461, 0.5249

5.4.2 Ovarian Cancer Detection

The data set comprises of 162 serum profiles from ovarian cancer patients, and 91 serum profiles from normal subjects who do not have cancer. Each serum profile comprises mass-to-charge ratios (*M/Z* values) of intensities that range from 0.0000786 to 19995.513 [163].

To ensure comparability across the spectra, the *M/Z* intensity values are normalized according to the procedure that has been recommended by the Clinical Proteomics Program Databank and described by Sorace and Zhan in [163]. The normalization is done over all the 253 examples for all the available *M/Z* identities using the following formula:

$$NI = \frac{RI - Min}{Max - Min} \quad (\text{Eq. 5.1})$$

where *NI* represents the normalized intensity, *RI* represents the raw intensity, and *Min* and *Max* refer to the minimum and the maximum intensity among the pooled examples, respectively. After the normalization, the *M/Z* intensity values will be between 0 and 1.

Similar to Sorace and Zhan [163], we randomly select 81 ovarian cancer patients and 45 control subjects for training, leaving the rest for testing. Figure 5.5 presents the Bayesian net-

Figure 5.5: BN Learned From the Ovarian Cancer Training Data.

Table 5.3: The Discovered Error Information on the Proteins. The Proteins are Listed in Decreasing Order of Estimated Error Rate.

Name of the Protein (M/Z)	The Estimated Error Rate
MZ244.66041	0.2063
MZ434.29682	0.1429
MZ261.88643	0.0952
MZ262.18857	0.0952
MZ435.07512	0.0476
MZ245.24466	0.0079
MZ244.95245	0.0079
MZ246.12233	0.0079
MZ245.53704	0.0
MZ245.8296	0.0

work (BN) that has been automatically learned from the training data based on the top 10 proteins with the highest information gains (Table 5.2). The *Markov blanket* for the target node of *Category* has been demarcated. It is interesting to note that the proteins in the learned *Markov blanket* overlap with those that Sorace and Zhan [163] identified via stepwise discriminant analysis. Specifically, “MZ261.88643” and “MZ435.07512” are found to have exceptionally strong diagnostic power in both experiments. Based on this, there is further reason to believe that these proteins are closely related to ovarian cancer.

Using this model to predict on the test data without applying noise correction yields reasonably accurate results. It misclassifies just one of the 81 cancer and one of the 46 normal samples, giving a sensitivity (cancer-class recall) of 98.77%, a specificity (normal-class recall) of 97.83%, and a positive predictive value (cancer-class precision) of 98.77%.

We investigate whether the misclassifications can be corrected based on our proposed knowledge discovery process. Using a minimum error threshold of zero, our Procedure 8

Table 5.4: Examples that are Correctly Classified Only After Applying Our Noise Correction Procedure. The Proteins are Listed in the Same Order As Shown in Table 5.2.

Protein Values Before Error Compensation	Class Label	Prediction (Prob.)
2, 0, 1, 1, 0, 0, 2, 0, 1, 2	Cancer	Normal (0.600)
0, 0, 1, 1, 1, 0, 1, 1, 0, 0	Normal	Cancer (0.564)
Protein Values After Error Compensation	Class Label	Prediction (Prob.)
2, 0, 1, 1, 0, 0, 0, 0, 1, 2	Cancer	Cancer (0.832)
0, 0, 1, 1, 1, 0, 1, 1, 0, 0	Normal	Normal (0.619)

automatically discovers that eight out of the ten proteins possess some evidence of errors (Table 5.3), although only the first few most-erroneous proteins have an estimated error rate that is above 0.10. This information about the individual protein's error probability encompasses the important discovered knowledge about each measurement's reliability.

Next, using the learned model presented in Figure 5.5, we apply our proposed process to the same test examples. Based on the estimated error rates, we compute the likelihoods estimates for each protein from its values in the test cases and input these into the BN. This produces a perfect classification for all 127 unseen samples. Details of the corrections in the two examples that are misclassified earlier are presented in Table 5.4. We can see that the beliefs of proteins within the Bayesian network have been updated, to the extent that the most-probable protein values may differ from their findings. This results in the correct predictions, each of which has a higher probability, or a higher level of certainty.

We repeat the experiment for five rounds of ten-fold stratified validations. In each round, the data set is randomly partitioned into ten equal-sized parts. One part is left out for testing, while we train on the remaining samples, and we repeat this for each of the ten parts to get the average accuracy. Based on just the learned BNs, the predictive accuracy, i.e., the average percentage of test samples that are correctly classified within the fifty splits, is 96.46%. We note that this already corresponds to just a single misclassification on average. Our proposed noise correction process improves the BN accuracy further to 96.70%, with more of the splits producing perfect diagnostic sensitivity and specificity.

Table 5.5: EBI Explanations for the First Test Example in Table 5.4.

(a) Explanation for the Diagnosis.

*BN predicts Cancer ($p=0.832$), as
 $MZ245.24466 = 1$ ($p=1.0$),
 $MZ245.8296 = 0$ ($p=1.0$),
 $MZ244.66041 = 0$ ($p=0.6$),
 $MZ435.07512 = 1$ ($p=1.0$), and
 $MZ261.88643 = 2$ ($p=0.6$).*

(b) Markov Value Correction During the Inference.

*MZ244.66041 is corrected from 2 to 0, as
 Given $MZ245.8296 = 0$,
 ($MZ244.66041 = 0$) has $p=0.6$,
 while ($MZ244.66041 = 2$) only has $p=0.3$.*

Apart from predictive accuracy, the direct interpretability of the discovered model of causal protein interactions is an important advantage of our proposed BN-based process. For instance, besides the information on *Markov blanket*, the process provides insights to the effects of errors in different proteins upon the diagnosis. With reference to Figure 5.5, protein “MZ244.66041” is directly connected to the target node “Category”, as well as to protein “MZ245.8296”. From Table 5.3, these are respectively the most and the least erroneous proteins. Our procedure would have thus entered the value of “MZ244.66041” with a much lower certainty than “MZ245.8296”, allowing the latter to correct the former. True enough, we see from the first example of Table 5.4 that a key reason for the correct prediction after error compensation is the correction in value of “MZ244.66041”. Table 5.5 shows the EBI explanation for this BN diagnosis. This automatically-generated explanation clearly highlights the compensation for “MZ244.66041” during the diagnosis.

5.4.3 Acute Leukemia Subtype Classification

5.4.3.1 The ALL/AML Data Set

The first data set classifies tumor samples into Acute Lymphoblastic Leukemia (ALL) and Acute Myeloid Leukemia (AML) based on their gene expressions [168]. The data set comprises 38 training cases (27 ALL, 11 AML) and 34 testing cases (20 ALL, 14 AML).

Table 5.6 shows that our method generates just a single misclassification, which outperforms prior results on this data set including weighted voting, support vector machine (SVM),

Table 5.6: Test Accuracy in the ALL/AML Data Set.

Method	Number of Genes	Errors
Weighted Voting	50	5
SVM	25	2
EP	1	3
ARAM	10	2
BN-NC	24	1

Figure 5.6: Markov Blankets in BNs Learned from the ALL/AML Data.

emerging patterns (EP), and adaptive resonance associative map (ARAM) [170].

Figure 5.6.a shows the Markov blanket of the learned BN. Specifically, genes *Zyxin* and *CST3* *Cystatin C* are observed to have a high information gain of 0.9 and 0.7 respectively. No error is detected for both genes in our process, which helps to explain the BN’s ability to discriminate the subtypes even with no noise correction. To generate explanations for noise correction using EBI, we reduce the number of genes to 18. BN learning based on these genes displaces *CST3* *Cystatin C* from the Markov blanket (Figure 5.6.b), resulting in as many as 5 misclassifications on the same test set. We discover an error rate of 0.08 for both ADM Adrenomedullin and ELA2 Elastatse 2, neutrophil. Our proposed knowledge discovery process with noise correction improves the BN accuracy to 3 misclassifications.

Table 5.7 shows the EBI explanations generated for a test case, before and after noise correction. The rules present the user with important insights to the inference. First, the BN re-

Table 5.7: EBI Explanations for an ALL/AML Test Example.

(a) Without Noise Correction.	(b) After Noise Correction.
<i>Class is AML</i> ($p=0.517$), as <i>Zyxin</i> ≤ 994 ($p=1.0$), <i>ADM</i> > 185 ($p=1.0$), and <i>ELA2</i> > 197.5 ($p=1.0$).	<i>Class is ALL</i> ($p=0.639$), as <i>Zyxin</i> ≤ 994 ($p=1.0$), <i>ADM</i> > 185 ($p=0.8$), and <i>ELA2</i> > 197.5 ($p=0.8$).

Table 5.8: Test Accuracy in the Pediatric ALL Data Set. The Best Results for Each of the Six Subtypes are Indicated in Bold.

Subtype	k -NN	BN	BN-NC
T-ALL	100	100	100
E2A-PBX1	100	99	100
TEL-AML1	100	99	100
BCR-ABL	97	98	99
MLL	100	100	100
Hype.>50	98	96	96

verses its diagnosis after accounting for uncertainty in the Markov values. Second, this change in tack is due to the lower beliefs in values of genes ADM Adrenomedullin and ELA2 Elastase 2. Finally, BN aided by noise correction predicts ALL with a higher confidence than it predicted AML, which matches the fact that this is an ALL example.

5.4.3.2 The Pediatric ALL Data Set

The second data set classifies 327 Acute Lymphoblastic Leukemia (ALL) tumor samples from children into six subtypes based on their gene expressions [169]. The training set comprises 215 examples, and another 112 samples are provided for testing. Depending on which one of the six acute leukemia subtypes is to be classified, an example in the data set is either labelled as “belonging” or “not belonging” to that ALL subtype [169].

Table 5.8 compares the performance of our process against the best results from [169], which uses the k -Nearest Neighbor (k -NN) for prediction. Our approach outperforms BN on subtypes E2A-PBX1, TEL-AML1 and also BCR-ABL. Due to the relative ease of the classification task, it is probably too close to conclude whether k -NN or our process has performed better on this problem. Nevertheless, we outperform all the other approaches reported in [169], including that of support vector machine and artificial neural network.

Tables 5.9(a) and 5.9(b) show the generated EBI explanations for the BN inference on a test example of TEL-AML1, before and after applying our proposed noise correction procedure. Similar to what we have seen earlier in the case of ALL/AML, noise correction enables the BN to consider the uncertainty in the gene measurements, to correctly classify this case as “not belonging” to TEL (not_TEL). In addition, the explanations in Table 5.9(c) provide users with insight to the Markov value corrections during inference.

Table 5.9: EBI Explanations for a TEL-AML1 Test Example.

(a) Without Noise Correction.	(b) After Noise Correction.
<i>Class is TEL ($p=0.773$), as</i>	<i>Class is not_TEL ($p=0.919$), as</i>
<i>a38652 = 0 ($p=1.0$),</i>	<i>a38652 = 0 ($p=1.0$),</i>
<i>a36239 = 2 ($p=1.0$),</i>	<i>a36239 = 2 ($p=0.6$),</i>
<i>a1077 = 1 ($p=1.0$),</i>	<i>a1077 = 1 ($p=0.8$),</i>
<i>a38203 = 0 ($p=1.0$),</i>	<i>a38203 = 0 ($p=1.0$),</i>
<i>a35614 = 0 ($p=1.0$),</i>	<i>a35614 = 0 ($p=1.0$),</i>
<i>a32224 = 0 ($p=1.0$),</i>	<i>a32224 = 0 ($p=1.0$),</i>
<i>a37780 = 1 ($p=1.0$),</i>	<i>a37780 = 0 ($p=0.7$),</i>
<i>a38578 = 1 ($p=1.0$),</i>	<i>a38578 = 1 ($p=0.7$),</i>
<i>a41442 = 0 ($p=1.0$),</i>	<i>a41442 = 0 ($p=1.0$),</i>
<i>a36985 = 1 ($p=1.0$), and</i>	<i>a36985 = 1 ($p=0.6$), and</i>
<i>a1299 = 2 ($p=1.0$).</i>	<i>a1299 = 0 ($p=0.8$).</i>
<hr/>	
(c) Markov Value Corrections During the Inference.	
<i>a37780 is corrected from ($a37780 = 1$) to ($a37780 = 0$), as</i>	
<i>Given $a38203 = 0$ and $a35665 = 0$,</i>	
<i>($a37780 = 0$) has $p=0.7$, while ($a37780 = 1$) only has $p=0.3$.</i>	
<i>a1299 is corrected from ($a1299 = 2$) to ($a1299 = 0$), as</i>	
<i>Given $a35614 = 0$,</i>	
<i>($a1299 = 0$) has $p=0.8$, while ($a1299 = 2$) only has $p=0.1$.</i>	

As we have witnessed so far throughout the preceding series of experiments, in addition to a highly robust recommendation performance, the ability to generate interpretable knowledge is the key strength of our BN-based knowledge discovery approach. In assigning each unseen example to its most-probable category, the Bayesian network uses its probabilities as the basis for its confidence. Most importantly, the new knowledge that is learned from the data can be readily integrated with the prior knowledge from biologists and other experts in related fields. As an effective analytical tool, we expect the discovered error information about the genotypic features to be particularly useful, e.g. in improving aspects of the data collection process.

5.5 Chapter Summary

A key challenge that faces bio-data classification, e.g. in cancer detection applications, is the presence of noise in the feature values that lowers classification accuracy. Undetected errors in measurements from protein spectroscopy and gene microarray experiments can inadvertently lead to misdiagnoses with serious consequences. We observe that many data sets contain dependencies between features, which means that there are input redundancies in the data that can be exploited for noise correction. In addition to a higher resilience against errors, diagnostic

CHAPTER 5. AN EFFECTIVE KNOWLEDGE DISCOVERY PROCESS FOR NOISY BIO-DATA CLASSIFICATIONS

models should also be more comprehensible so that a deeper understanding of the dependencies among biological entities such as proteins and genes may become possible. In this chapter, we present a novel knowledge discovery process that effectively addresses these challenges.

The proposed knowledge discovery process consists of four steps: data preprocessing, feature dependencies modelling, noise discovery, and prediction with explanation. After identifying the salient features, we capture their dependencies in a Bayesian network (BN). Using the BN to predict each feature in turn, we estimate the features' error rates based on the proportion of mismatches in their stated and predicted values. The error rates are then used to compute likelihoods for the uncertain feature values during classification, so that errors in the features' stated values can be effectively corrected by the BN classifier. In complement to each prediction, our EBI procedure automatically generates a rule that explains the classification variable using the feature values in its Markov blanket. The procedure is then applied recursively to in turn explain those feature values that are missing or corrected during the classification.

We validate the proposed knowledge discovery process through a series of experiments, involving HIV-1 drug resistance prediction based on amino acid mutations, ovarian cancer detection based on protein mass-to-charge intensities, and acute leukemia subtype classification based on gene expression data sets. The results from the validations show that the proposed process is effective in correcting noisy input values, which translates to higher classification accuracy for the BN classifier. In contrast to the incomprehensible "black-box" nature of other prediction models, the BN offers an interpretable graphical model of the dependencies among the salient features. Moreover, the classification rules that are generated by our EBI procedure provide clear explanations for each of the BN classification and noise correction decisions.

Chapter 6

Conclusions

In this thesis, we have highlighted a number of major challenges that need to be addressed before context information can be effectively used to improve the performance of recommendation systems. Firstly, the limited resources in context-aware recommender systems necessitate the identification of the minimal set of important contexts for each particular user, so that only the necessary resources would be expended in acquiring those contexts. Secondly, the inherent uncertainties in the acquired context values make it critical for the recommenders to effectively compensate for noise in the form of frequently missing and erroneous context inputs. Finally, in addition to recommendation accuracy, the predictions generated from the system have to be clearly explained so that they would be more readily accepted by the users. The work that we have presented in this dissertation report effectively addresses all of the above challenges.

6.1 Recap Of Contributions

We have introduced a novel Bayesian network (BN)-based approach to develop context-aware recommender systems that systematically addresses the aforementioned challenges. The following are the specific research contributions presented in this thesis:

- **Learning the Minimal Set of Contexts**

We have introduced an intuitive and highly reliable learning procedure that automatically discovers the minimal set of important contexts for a specific user and recommendation task. Experiments on a restaurant recommender show that the proposed procedure accurately discovers context dependencies to yield a clear graphical model of the problem, and we are able to consistently identify the minimal set of important contexts particular to individual users.

- **Handling Missing and Erroneous Context Inputs for Accurate Prediction**

We have introduced a two-tiered context model and a novel error compensation procedure which effectively capture context dependencies to compensate noisy context inputs. Results from experiments on a restaurant recommender and a web page recommendation data set show that the proposed procedure effectively harnesses the context dependencies in the learned BN to achieve significantly more robust predictions under missing or erroneous context inputs.

- **Explaining Recommendations in Bayesian Networks**

We have introduced a novel Explaining BN Inferences (EBI) procedure that explains the BN recommendations and their corresponding noise compensations. Results from mushrooms classification, water purification and web page recommendation problems show that EBI produces concise and comprehensible explanations for BN noise compensations, to promote user appreciation of the recommendations generated by the BN-based recommender systems.

- **An Effective Knowledge Discovery Process for Noisy Bio-Data Classifications**

In addition, we have introduced an effective knowledge discovery process for bio-data classifications that automatically corrects noisy inputs and explains both the classifications and the noise corrections. Results from experiments on HIV-1 drug resistance prediction, ovarian cancer detection, and acute leukemia subtype classification show that noise correction significantly improves the BN's accuracy, and that the EBI procedure generates meaningful explanations which involve features with salient biological significance. While the process is validated using biological domains, it can be readily and effectively used for noisy domains in general.

6.2 Limitations and Future Work

In this dissertation report, we have presented a set of novel and effective solutions that have enabled us to reach our stated objectives. Nevertheless, as with all evolving research doctrines, there remain certain opportunities for enhancement in future work.

- **Modification to the Minimal Set Learning Procedure**

In each iteration of the proposed minimal context set learning procedure, we retain those parameters that are directly connected to the target within the learned BN, i.e., just the parents

CHAPTER 6. CONCLUSIONS

and children of the target node are retained. This has been demonstrated to be highly effective for identifying the minimal context set of each specific user. A possible modification to the current procedure is to also retain the target's spouses, i.e., the other parent nodes of its children, so that we cover all the nodes in the target's Markov blanket.

While the Markov blanket of the target variable contains the entire set of strongly-relevant features that completely predicts the target's value (we have exploited this property in our EBI explanation procedure), it is also known in classical feature selection that not all the strongly-relevant features are *necessary* for accurate prediction. A disadvantage of using all the Markov blanket nodes is that the number of spouse nodes, each of which remains *d-separated* from the target node until their common child is instantiated, can add significantly to the network size, which remains a major burden for existing BN learning methods. While our current findings have demonstrated that the minimal set of contexts can be quickly and accurately identified among the directly connected nodes, future experiments can be designed to ascertain whether analyzing the potentially much larger set of Markov blanket parameters may be more effective for certain problems. This is a straightforward modification to our already working procedure and may be regarded as more of an alternative version than an enhancement of our approach.

- **Adapting the Bayesian Network**

In the current BN-based recommendation approach, the BN employed for recommendation is learned from observation data that are presented as a single batch of training examples to the BN learning algorithm. When a substantial number of new or additional examples become available to the system over time, the structure or topology of the BN has to be updated by learning on both the original examples as well as the additional examples. To the best of our knowledge, none of the existing BN learning solutions is able to dynamically *adapt the BN's structure* according to new examples, although there already exist techniques for dynamically revising the BN's encoded conditional probabilities or *beliefs* based on incoming examples.

In the case of belief revision, we can improve prediction accuracy by making the BN learn from the last correct answer as soon as the user's choice is known for that prediction. In other words, each time after the learned BN makes a recommendation, which is a best-effort prediction since the BN has no prior knowledge of the correct answer, we feed back into the network

that same example along with the correct answer. We then instruct the BN to revise its learned conditional probabilities based on this new “training” example. We have compared the prediction performance with and without this improvisation learning using a number of networks, and the results show that the average accuracy for each test data set has either been maintained (if the accuracy is already close to 100%), or has increased noticeably after allowing for belief revisions. This case-by-case learning strategy represents an adaptive learning approach similar to those of the class of self-organizing neural networks known as adaptive resonance theory networks (ART) [170]. We envision that as the current technology for automatic BN learning from data advances, it would one day become possible to dynamically improve the prediction performance further by also adapting the topology of the learned BN in an incremental manner.

- **Customizing the EBI Explanations**

The EBI procedure that we have introduced can be further customized to generate explanations that are specially suited for particular applications and users. Besides the context-specific independence relations among variables, which have been known to exist in many real-world applications [126, 132–134], the EBI procedure can be modified to exploit additional types of variable interactions when explaining the BN inferences in specialized domains. For instance, the less prevalent *causal independence* relation [175], which is defined as a situation where multiple causes contribute independently to a common effect, has known applicability to the class of noisy OR-gate problems [17]. Likewise, while the EBI procedure currently uses decision trees to capture the interactions among the Markov variables and decision tree rules to represent the resulting explanations, other representations like M-of-N rules [176] may provide richer semantics for specific problems. In addition to customizing the EBI explanations to particular domains, some users may prefer to be presented with explanations that are formulated differently. While our research results have demonstrated the general feasibility of EBI, the collection of user evaluations for customization is an important follow-on work.

- **Collaborating with Domain Experts**

Having a systematic knowledge discovery process to learn and exploit genotypic feature dependencies for robust and comprehensible recommendations within the biological domain,

CHAPTER 6. CONCLUSIONS

a logical next step is to work with the domain experts to interpret and validate the discovered biological knowledge. In this way, we can build a classifier that not only is extremely sensitive and specific, but also describes biologically plausible feature interactions, thereby allowing the knowledge that is found from small data sets to generalize well. Hopefully, this can contribute to better screening tools, e.g. for women who are at high risk of ovarian cancer, or children with predisposition to acute leukemia. As our solutions are applicable to all noisy recommendation problems, the same process can also be readily applied to domains other than bio-classification. We can then collaborate with the experts in each domain to, for example, improve the handling of those domain-specific context features which our process finds to be suffering from noise.

References

- [1] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.
- [2] J. Schafer, J. Konstan, and J. Riedl. Electronic commerce recommender applications. *Journal of Data Mining and Knowledge Discovery*, 5(1-2):115–152, 2000.
- [3] B. Xiao, E. Aimeur, and J. Fernandez. PCFinder: An intelligent product recommendation agent for e-commerce. In *Proceedings of IEEE International Conference on E-Commerce (CEC)*, pages 181–188, 2003.
- [4] J. Ji, C. Liu, J. Yan, and N. Zhong. Bayesian networks structure learning and its application to personalized recommendation in a B2C portal. In *Proceedings of International Conference on Web Intelligence (WI)*, pages 179–184, Beijing, China, September 2004. IEEE Computer Society Press.
- [5] H.-W. Tung and V.-W. Soo. A personalized restaurant recommender agent for mobile e-service. In *Proceedings of IEEE International Conference on e-Technology, e-Commerce, and e-Services (EEE)*, pages 259–262. IEEE Computer Society, 2004.
- [6] A. Felfernig and A. Kiener. Knowledge-based interactive selling of financial services using FSAdvisor. In *Proceedings of Innovative Applications of Artificial Intelligence Conference (IAAI)*, pages 1475–1482, Pittsburgh, Pennsylvania, 2005. AAAI Press.
- [7] D. Goren-Bar and T. Kuflik. Don’t Miss-R - Recommending restaurants through an adaptive mobile system. In *Proceedings of International Conference on Intelligent User Interface (IUI)*, pages 250–252, Funchal, Madeira, Portugal, 2004. ACM Press.
- [8] M. van Setten, S. Pokraev, and J. Koolwaaij. Context-aware recommendations in the mobile tourist application COMPASS. In *Proceedings of International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH)*, pages 235–244, Eindhoven University of Technology, The Netherlands, August 2004.
- [9] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions On Information Systems (TOIS)*, 23(1):103–145, 2005.

REFERENCES

- [10] O. Madani and D. DeCoste. Contextual recommender problems. In *Proceedings of KDD Workshop on Utility-Based Data Mining (UBDM)*, pages 86–89, 2005.
- [11] K. Henriksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. In *Proceedings of International Conference on Pervasive Computing*, pages 167–180. Springer-Verlag, 2002.
- [12] C. K. Hess and R. H. Campbell. An application of a context-aware file system. *Personal and Ubiquitous Computing*, 7(6):339–352, 2003.
- [13] N. Kern, B. Schiele, H. Junker, P. Lukowicz, and G. Troster. Wearable sensing to annotate meeting recordings. *Personal and Ubiquitous Computing*, 7(5):263–274, October 2003.
- [14] J. Trevor, D. M. Hilbert, D. Billsus, J. Vaughan, and Q. T. Tran. Contextual contact retrieval. In *Proceedings of International Conference on Intelligent User Interface (IUI)*, pages 337–339, Funchal, Madeira, Portugal, 2004. ACM Press.
- [15] L. Terrenghi and A. Zimmermann. Tailored audio augmented environment for museums. In *Proceedings of International Conference on Intelligent User Interface (IUI)*, pages 334–336, Funchal, Madeira, Portugal, 2004. ACM Press.
- [16] G.-E. Yap, A.-H. Tan, and H.-H. Pang. Dynamically-optimized context in recommender systems. In *Proceedings of International Conference on Mobile Data Management (MDM)*, pages 265–272, Ayia Napa, Cyprus, May 2005. ACM Press.
- [17] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.
- [18] G.-E. Yap, A.-H. Tan, and H.-H. Pang. Discovering causal dependencies in mobile context-aware recommenders. In *Proceedings of International Conference on Mobile Data Management (MDM)*, pages 4 (in CD-ROM Proceedings), Nara, Japan, May 2006. IEEE Computer Society Press.
- [19] G.-E. Yap, A.-H. Tan, and H.-H. Pang. Discovering and exploiting causal dependencies for robust mobile context-aware recommenders. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(7):977–992, July 2007.
- [20] G.-E. Yap, A.-H. Tan, and H.-H. Pang. Explaining inferences in Bayesian networks. *Applied Intelligence (Online First)*, October 2007. doi: 10.1007/s10489-007-0093-8.
- [21] G.-E. Yap, A.-H. Tan, and H.-H. Pang. Learning causal models for noisy biological data mining: An application to ovarian cancer detection. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 354–359, Vancouver, British Columbia, July 2007. AAAI Press.

REFERENCES

- [22] B. Schilit and M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, 1994.
- [23] A. Schmidt and K. van Laerhoven. How to build smart appliances. *IEEE Personal Communications*, 8(4):66–71, August 2001.
- [24] R. Hull, P. Neaves, and J. Bedford-Roberts. Towards situated computing. In *Proceedings of International Symposium on Wearable Computers (ISWC)*, pages 146–153, Cambridge, MA, USA, October 1997.
- [25] P. J. Brown. The stick-e document: A framework for creating context-aware applications. In *Proceedings of Electronic Publishing*, pages 259–272, Palo Alto, 1996.
- [26] N. Ryan, J. Pascoe, and D. Morse. Enhanced reality fieldwork: The context-aware archaeological assistant. In *Proceedings of Computer Applications in Archaeology (CAA)*, University of Birmingham, April 1997. Archaeopress (CD-ROM).
- [27] A. K. Dey. Context-aware computing: The CyberDesk project. In *Proceedings of AAAI Spring Symposium on Intelligent Environments*, pages 51–54, Menlo Park, CA, 1998.
- [28] A. K. Dey and G. D. Abowd. Towards a better understanding of context and context-awareness. Technical Report GIT-GVU-99-22, June 1999. Panel paper at International Symposium on Handheld and Ubiquitous Computing (HUC 1999).
- [29] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 85–90, Santa Cruz, CA, US, December 1994. IEEE Computer Society Press.
- [30] A. K. Dey and G. D. Abowd. A conceptual framework and a toolkit for supporting rapid prototyping of context-aware applications. *Human-Computer Interactions Journal (HCI)*, 16(2-4):7–166, 2001.
- [31] J. W. Koolwaaij and P. Strating. Service frameworks for mobile context-aware applications. In *Proceedings of eChallenges Workshop on Future Workplaces: Supporting Mobile User and Worker*, Palazzo Re Enzo, Bologna, Italy, October 2003.
- [32] P. Dourish. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8(1):19–30, 2004.
- [33] Google. Google SMS - Text message Google for quick results. Retrieved on 3rd January 2008 from http://www.google.com/intl/en_us/mobile/sms/index.html.
- [34] The Weather Co. SMS Weather - Weather via SMS from Weatherzone. Retrieved on 3rd January 2008 from <http://www.weatherzone.com.au/services/smsweather.jsp>.

REFERENCES

- [35] Smarter.com (a MeziMedia Site). Smarter.SMS - Check prices no matter where you are. Retrieved on 3rd January 2008 from <http://www.smarter.com/sms/>.
- [36] C. Steinfield. The development of location based services in mobile commerce. In *E-life After the Dot Com Bust*, pages 177–197. 2004.
- [37] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC)*, 2(4):263–277, 2007.
- [38] T. Kindberg, J. J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic. People, places, things: Web presence for the real world. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 19–28, Monterey, CA, USA, December 2000. IEEE Computer Society Press.
- [39] D. Salber, A. K. Dey, and G. D. Abowd. The Context Toolkit: Aiding the development of context-enabled applications. In *Proceedings of SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 434–441, Pittsburgh, Pennsylvania, United States, 1999. ACM Press.
- [40] R. Power. Topic maps for context management. In *Proceedings of International Symposium on Information and Communication Technologies (ISICT)*, pages 199–204, Dublin, Ireland, 2003. Trinity College Dublin.
- [41] A. Ranganathan and R. H. Campbell. An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing*, 7(6):353–364, 2003.
- [42] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang. An ontology-based context model in intelligent environments. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, pages 270–275, San Diego, California, USA, January 2004.
- [43] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *Proceedings of Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004*, Nottingham, England, September 2004.
- [44] P. D. Turney. Robust classification with context-sensitive features. In *Proceedings of International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE)*, pages 268–276, Edinburgh, Scotland, 1993.
- [45] P. D. Turney. The identification of context-sensitive features: A formal definition of context for concept learning. In *Proceedings of ICML Workshop on Learning in Context-Sensitive Domains*, pages 53–59, Bari, Italy, 1996.

REFERENCES

- [46] P. D. Turney. The management of context-sensitive features: A review of strategies. In *Proceedings of ICML Workshop on Learning in Context-Sensitive Domains*, pages 60–66, Bari, Italy, 1996.
- [47] H. Wu, M. Siegel, and S. Abalay. Sensor fusion for context understanding. In *Proceedings of IEEE Instrumentation and Measurement Technology Conference (IMTC)*, Anchorage, AK, USA, May 2002.
- [48] H. Wu, M. Siegel, R. Stiefelhausen, and J. Yang. Sensor fusion using Dempster-Shafer Theory. In *Proceedings of IEEE Instrumentation and Measurement Technology Conference (IMTC)*, Anchorage, AK, USA, May 2002.
- [49] H. Wu, M. Siegel, and S. Abalay. Sensor fusion using Dempster-Shafer Theory II: Static weighting and Kalman Filter-like dynamic weighting. In *Proceedings of IEEE Instrumentation and Measurement Technology Conference (IMTC)*, Vail, CO USA, May 2003.
- [50] D. Guan, W. Yuan, S. Lee, and Y.-K. Lee. Context selection and reasoning in ubiquitous computing. In *Proceedings of 2007 Int. Conf. on Intelligent Pervasive Computing (IPC)*, pages 184–187, October 2007.
- [51] J. Pascoe. Adding generic contextual capabilities to wearable computers. In *Proceedings of IEEE International Symposium on Wearable Computers (ISWC)*, pages 92–99. IEEE Computer Society Press, 1998.
- [52] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992.
- [53] F. Hohl, U. Kubach, A. Leonhardi, K. Rothermel, and M. Schwehm. Next century challenges: Nexus - An open global infrastructure for spatial-aware applications. In *Proceedings of Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 249–255, Seattle, Washington, United States, 1999. ACM Press.
- [54] T. Hori and K. Aizawa. Context-based video retrieval system for the life-log applications. In *Proceedings of ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR)*, pages 31–38, Berkeley, California, 2003. ACM Press.
- [55] R. Hexel, C. Johnson, B. Kummerfeld, and A. J. Quigley. “Powerpoint to the People”: Suiting the word to the audience. In A. Cockburn, editor, *Proceedings of Australasian User Interface Conference (AUIC)*, volume 28 of *CRPIT*, pages 49–56, Dunedin, New Zealand, 2004. Australian Computer Society, Inc.

REFERENCES

- [56] L. Romero and N. Correia. HyperReal: A hypermedia model for mixed reality. In *Proceedings of ACM Conference on Hypertext and Hypermedia (HYPERTEXT)*, pages 2–9, Nottingham, UK, 2003. ACM Press.
- [57] J. E. Bardram. Applications of context-aware computing in hospital work - Examples and design principles. In *Proceedings of ACM Symposium on Applied Computing (SAC)*, pages 1574–1579, Nicosia, Cyprus, 2004. ACM Press.
- [58] J. D. Carswell, K. Gardiner, and M. Neumann. Wireless spatio-semantic transactions on multimedia datasets. In *Proceedings of ACM Symposium on Applied Computing (SAC)*, pages 1201–1205, Nicosia, Cyprus, 2004. ACM Press.
- [59] K. Fujinami, T. Yamabe, and T. Nakajima. “Take me with you!”: A case study of context-aware application integrating cyber and physical space. In *Proceedings of ACM Symposium on Applied Computing (SAC)*, pages 1607–1614, Nicosia, Cyprus, 2004. ACM Press.
- [60] M. Hatala, L. Kalantari, R. Wakkary, and K. Newby. Ontology and rule based retrieval of sound objects in augmented audio reality system for museum visitors. In *Proceedings of ACM Symposium on Applied Computing (SAC)*, pages 1045–1050, Nicosia, Cyprus, 2004. ACM Press.
- [61] M. Korkea-aho. Context-aware applications survey. Retrieved on 23th June 2005 from <http://www.hut.fi/~mkorkeaa/doc/context-aware.html>, April 2000.
- [62] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, NH, November 2000. URL <ftp://ftp.cs.dartmouth.edu/TR/TR2000-381.ps.Z>.
- [63] A. K. Dey, D. Salber, G. D. Abowd, and M. Futakawa. The Conference Assistant: Combining context-awareness with wearable computing. In *Proceedings of International Symposium on Wearable Computers (ISWC)*, page 21. IEEE Computer Society, 1999.
- [64] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. In *Proceedings of Int. World Wide Web Conference (WWW)*, pages 406–414, Hong Kong, 2001.
- [65] K. B. Korb and A. E. Nicholson. *Bayesian Artificial Intelligence*. CRC Press, 2003.
- [66] T. Gu, H. K. Pung, and D. Q. Zhang. A Bayesian approach for dealing with uncertain contexts. In *Proceedings of Conference on Pervasive Computing (Pervasive)*, “Advances in Pervasive Computing”, volume 176, Vienna, Austria, April 2004. Austrian Computer Society.

REFERENCES

-
- [67] G. Lacey and S. MacNamara. Context-aware shared control of a robot mobility aid for the elderly blind. *International Journal of Robotics Research*, 19(11):1054–1065, November 2000.
 - [68] G. Adomavicius and A. Tuzhilin. Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(6):734–749, June 2005.
 - [69] D. Goldberg, D. Nicholas, B. M. Oki, and D. B. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM (CACM)*, 35(12):61–70, 1992.
 - [70] M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM (CACM)*, 40(3):66–72, March 1997.
 - [71] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of International World Wide Web Conference (WWW)*, pages 285–295. ACM Press, 2001.
 - [72] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction (UMUAI)*, 12:331–370, 2002.
 - [73] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing Industry Report*, pages 76–80, 2003.
 - [74] Q. Li, B. M. Kim, D. H. Guan, and D. whan Oh. A music recommender based on audio features. In *Proceedings of ACM Special Interest Group on Information Retrieval (SIGIR)*, pages 532–533, Sheddield, South Yorkshire, UK, July 2004.
 - [75] M. Garden and G. Dudek. Semantic feedback for hybrid recommendations in Recommendz. In *Proceedings of IEEE International Conference on e-Technology, e-Commerce, and e-Services (EEE)*, pages 754–759, 2005.
 - [76] M. Garden and G. Dudek. Mixed collaborative and content-based filtering with user-contributed semantic features. In *Artificial Intelligence and the Web, a special track of AAAI-06*, Boston, Massachusetts, USA, July 2006. AAAI Press.
 - [77] D. Billsus and M. Pazzani. User modeling for adaptive news access. *User-Modeling and User-Adapted Interaction*, 10(2-3):147–180, 2000.
 - [78] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM (CACM)*, 40(3):77–87, 1997.

REFERENCES

- [79] GroupLens Research Project. MovieLens web-based movie recommender system. Retrieved on 13th February 2008 from <http://www.grouplens.org/>.
- [80] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM (CACM)*, 35(12):29–38, 1992.
- [81] S. E. Middleton, N. R. Shadbolt, and D. C. D. Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88, January 2004.
- [82] K. Lang. Newsweeder: Learning to filter news. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 331–339, Lake Tahoe, CA, 1995.
- [83] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of ACM Conference on Digital Libraries (DL)*, pages 195–240, June 2000.
- [84] P. Cano, M. Koppenberger, and N. Wack. An industrial-strength content-based music recommendation system. In *Proceedings of ACM Special Interest Group on Information Retrieval (SIGIR)*, page 673, Salvador, Brazil, August 2005.
- [85] P. Cano, M. Koppenberger, and N. Wack. Content-based music audio recommendation. In *Proceedings of Annual ACM International Conference on Multimedia (MM)*, pages 211–212, Hilton, Singapore, 2005.
- [86] G. J. F. Jones and P. J. Brown. Context-aware retrieval for ubiquitous computing environments. In *Proceedings of the Mobile HCI Workshop on Mobile and Ubiquitous Information Access*, volume 2954 of *LNCS*, pages 227–243. Springer, 2003.
- [87] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 43–52, Madison, WI, 1998. Morgan Kaufman.
- [88] J. Ji, Z. Sha, C. Liu, and N. Zhong. Online recommendation based on customer shopping model in e-commerce. In *Proceedings of International Conference on Web Intelligence (WI)*, pages 68–74, Halifax, Canada, 2003. IEEE Computer Society.
- [89] C. Needham, J. R. Bradford, A. J. Bulpitt, and D. R. Westhead. Application of Bayesian networks to two classification problems in bioinformatics. In *LASR'05 Workshop: Quantitative Biology, Shape Analysis and Wavelets*, pages 87–90, 2005.
- [90] O. G. Troyanskaya, K. Dolinski, A. B. Owen, R. B. Altman, and D. Botstein. A Bayesian framework for combining heterogeneous data sources for gene function prediction (in *Saccharomyces cerevisiae*). In *Proceedings of National Academy of Sciences (PNAS)*, volume 100, pages 8348–8353, July 2003.

REFERENCES

- [91] C. Huttenhower and O. G. Troyanskaya. Bayesian data integration: A functional perspective. In *Proceedings of Computational Systems Bioinformatics Conference (CSB)*, pages 341–351, 2006.
- [92] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, January 2006.
- [93] A. Chen. Context-aware collaborative filtering system: Predicting the user’s preference in the ubiquitous computing environment. In *Proceedings of International Workshop on Location- and Context-Awareness (LoCA)*, pages 244–253, 2005.
- [94] H. J. Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, 178:37–51, 2008.
- [95] D. Heckerman. A tutorial on learning with Bayesian networks. In M. Jordan, editor, *Learning in Graphical Models*, Cambridge MA, 1999. MIT Press.
- [96] J. H. Friedman. On bias, variance, 0/1-loss, and the curse of dimensionality. *Data Mining and Knowledge Discovery (DMKD)*, 1(1):55–77, March 1997.
- [97] M. Singh and G. Provan. Efficient learning of selective Bayesian network classifiers. Technical Report MS-CIS-95-36, Computer and Information Science Department, University of Pennsylvania, Philadelphia, PA., November 1995.
- [98] C. S. Wallace and K. B. Korb. Learning linear causal models by MML sampling. In A. Gammerman, editor, *Causal Models and Intelligent Data Management*. Springer, 1999.
- [99] D. Zhang, X. H. Wang, and K. Hackbarth. OSGi based service infrastructure for context aware automotive telematics. In *Proceedings of IEEE Vehicular Technology Conference (VTC Spring 2004)*, pages 2957–2961, Milan, Italy, May 2004.
- [100] T. Olsson. *Bootstrapping and Decentralizing Recommender Systems*. Dissertation for the Degree of Licentiate of Technology in Computer Science, Computer Science Division, Dept. of Info. Tech., Uppsala University, Uppsala, Sweden, June 2003.
- [101] Norsys Software Corporation. Netica-J: Java version of the Netica API. Retrieved on 23th June 2005 from <http://www.norsys.com/netica-j.html>.
- [102] R. E. Neapolitan. *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*. John Wiley & Sons, New York, 1990.
- [103] D. B. Rubin. Multiple imputation after 18+ years (with discussion). *Journal of the American Statistical Association*, 91:473–520, 1996.

REFERENCES

- [104] P. D. Allison. *Missing Data*. Quantitative Applications in the Social Sciences. Sage Publications, Inc, August 2001.
- [105] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, Hoboken, New Jersey, 2002.
- [106] V. J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:85–126, 2004.
- [107] H. Xiong, G. Pandey, M. Steinbach, and V. Kumar. Enhancing data analysis with noise removal. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 18(3):304–319, March 2006.
- [108] C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research (JAIR)*, 11:131–167, 1999.
- [109] S. C. Kachigan. *Statistical Analysis*. Radius Press, 1986.
- [110] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, California, USA, 2nd edition, 2005.
- [111] L. R. Hope and K. B. Korb. A Bayesian metric for evaluating machine learning algorithms. In G. I. Webb and X. Yu, editors, *Proceedings of Australasian Artificial Intelligence Conference (AI)*, pages 991–997. Springer-Verlag, 2004.
- [112] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences Department, University of Wisconsin-Madison, 2005.
- [113] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.
- [114] M. J. Pazzani, J. Muramatsu, and D. Billsus. Syskill & Webert: Identifying interesting web sites. In *Proceedings of National Conference on Artificial Intelligence (AAAI/IAAI)*, pages 54–61, Portland, Oregon, USA, August 1996. AAAI Press.
- [115] G. Salton. *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice Hall, Englewood Cliffs, New Jersey, 1971.
- [116] M. J. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331, 1997.
- [117] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.
- [118] M. Minsky and S. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969.

REFERENCES

- [119] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
- [120] P. Godfrey, J. Grant, J. Gryz, and J. Minker. *Logics for databases and information systems*, chapter Integrity Constraints: Semantics and Applications, pages 265–306. Kluwer International Series In Engineering And Computer Science. Kluwer Academic Publishers, Norwell, MA, USA, 1998.
- [121] R. L. Teach and E. H. Shortliffe. An analysis of physician attitudes regarding computer-based clinical consultation systems. *Computers and Biomedical Research*, 14:542–558, 1981.
- [122] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 241–250, Philadelphia, USA, 2000. ACM Press.
- [123] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- [124] H. J. Suermondt. *Explanation in Bayesian Belief Networks*. PhD thesis, Medical Information Sciences, Stanford University, Palo Alto, California, March 1992.
- [125] P. Haddawy, J. Jacobson, and C. E. Kahn Jr. BANTER: A Bayesian network tutoring shell. *Artificial Intelligence in Medicine*, 10:177–200, 1997.
- [126] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 115–123, Reed College, Portland, Oregon, USA, August 1996. Morgan Kaufmann.
- [127] P. Humphreys. *The Chances of Explanation: Causal Explanation in the Social, Medical, and Physical Sciences*. Princeton University Press, Princeton, New Jersey, 1989.
- [128] M. Strevens. *The Macmillan Encyclopedia of Philosophy, 2nd Edition*, chapter Scientific Explanation. Macmillan Reference USA, 2006.
- [129] S. K. M. Wong and W. Ziarko. INFER - An adaptive decision support system based on the probabilistic approximate classification. In *Proceedings of Int. Workshop on Expert Systems and Their Applications*, pages 713–725, Avignon, 1986.
- [130] A. Tversky and D. Kahneman. Judgement under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131, September 1974.
- [131] N. Pennington and R. Hastie. Explanation-based decision making: Effects of memory structure on judgement. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 14(3):521–533, 1988.

REFERENCES

- [132] N. L. Zhang and D. Poole. On the role of context-specific independence in probabilistic inference. In *Proceedings of Int. Joint Conference on Artificial Intelligence (IJCAI)*, pages 1288–1293, Stockholm, Sweden, 1999. Morgan Kaufmann.
- [133] C. J. Butz. Exploiting contextual independencies in web search and user profiling. In *Proceedings of World Congress on Computational Intelligence (WCCI)*, volume 2, pages 1051–1056, Honolulu, Hawaii, USA, 2002.
- [134] D. Poole and N. L. Zhang. Exploiting contextual independence in probabilistic inference. *Journal of Artificial Intelligence Research (JAIR)*, 18:263–313, 2003.
- [135] C. Lacave and F. J. Diez. A review of explanation methods for Bayesian networks. *Knowledge Engineering Review*, 17:107–127, 2002.
- [136] U. Chajewska and D. L. Draper. Explaining predictions in Bayesian networks and influence diagrams. In *Proceedings of AAAI Spring Symposium*, Stanford University, Palo Alto, Ca., 1998.
- [137] Z. Zhou, H. Liu, S. Z. Li, and C. S. Chua. Rule mining with prior knowledge: A belief networks approach. *Intelligent Data Analysis*, 5(2):95–110, 2001.
- [138] M. Druzdzel and M. Henrion. Using scenarios to explain probabilistic inference. In *Working Notes of AAAI Workshop on Explanation*, pages 133–141, July 1990.
- [139] J. R. Koiter. *Visualizing Inference in Bayesian Networks*. PhD thesis, Delft University of Technology, Delft, The Netherlands, 2006.
- [140] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, 1986.
- [141] S. M. Olmsted. *On Representing and Solving Decision Problems*. PhD thesis, Dept. of Engineering-Economic Systems, Stanford University, Palo Alto, California, 1983.
- [142] N. L. Zhang. Inference in Bayesian networks: The role of context-specific independence. Technical Report HKUST-CS98-09, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, 1998.
- [143] P. Ibarguengoytia, L. Sucar, and S. Vadera. A probabilistic model for sensor validation. In *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 332–339, Reed College, USA, 1996. Morgan Kaufmann.
- [144] A. Asuncion and D. J. Newman. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2007. URL <http://www.ics.uci.edu/~mllearn/MLRepository.html>.

REFERENCES

- [145] F. V. Jensen, U. Kjærulff, K. G. Olesen, and J. Pedersen. An expert system for control of waste water treatment - A pilot project. Technical report, Judex Datasystemer A/S, Aalborg, Denmark, 1989. In Danish.
- [146] S. M. Weiss and N. Indurkha. Reduced complexity rule induction. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 678–684, Sydney, Australia, August 1991. Morgan Kaufmann.
- [147] P. Bajcsy, J. Han, L. Liu, and J. Yang. *Data Mining in Bioinformatics*, chapter Survey of Biodata Analysis from a Data Mining Perspective, pages 9–39. Advanced Information and Knowledge Processing. Springer Berlin Heidelberg, 2005.
- [148] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems (KIS)*, 14(1):1–37, December 2007.
- [149] S. Y. Rhee, J. G. Matthew, R. Kantor, B. J. Betts, J. Ravela, and R. W. Shafer. Human immunodeficiency virus reverse transcriptase and protease sequence database. *Nucleic Acids Research (NAR)*, 31(1):298–303, 2003.
- [150] H. Saigo, T. Uno, and K. Tsuda. Mining complex genotypic features for predicting HIV-1 drug resistance. *Bioinformatics*, 23(18):2455–2462, August 2007.
- [151] E. F. Petricoin, A. M. Ardekani, B. A. Hitt, P. J. Levine, V. A. Fusaro, S. M. Steinberg, G. B. Mills, C. Simone, D. A. Fishman, E. C. Kohn, and L. A. Liotta. Use of proteomic patterns in serum to identify ovarian cancer. *Lancet*, 359:572–577, 2002.
- [152] V. Kumar G. Pandey and M. Steinbach. Computational approaches for protein function prediction. Technical Report TR 06-028, Department of Computer Science and Engineering, University of Minnesota, October 2006.
- [153] N. Friedman, M. Linial, I. Nachman, and D. Pe’er. Using Bayesian networks to analyze expression data. In *Proceedings of International Conference on Computational Molecular Biology (RECOMB)*, pages 127–135, Tokyo, Japan, April 2000.
- [154] J. Han. How can data mining help bio-data analysis? (extended abstract). In M. J. Zaki, J. T. L. Wang, and H. T. T. Toivonen, editors, *Proceedings of BIODDD’02 Workshop on Data Mining in Bioinformatics (with SIGKDD’02)*, pages 1–2, 2002.
- [155] R. Yamaguchi and T. Higuchi. State-space approach with the maximum likelihood principle to identify the system generating time-course gene expression data of yeast. *Int. Journal of Data Mining and Bioinformatics (IJDMB)*, 1(1):77–87, 2006.

REFERENCES

- [156] S.-Y. Rhee, J. Taylor, G. Wadhera, A. Ben-Hur, D. L. Brutlag, and R. W. Shafer. Genotypic predictors of human immunodeficiency virus type 1 drug resistance. In *Proceedings of National Academy of Sciences (PNAS)*, volume 103, pages 17355–17360, November 2006.
- [157] K. Deforche, T. Silander, R. Camacho, Z. Grossman, M. A. Soares, K. Van Laethem, R. Kantor, Y. Moreau, and A.-M. Vandamme. Analysis of HIV-1 pol sequences using Bayesian networks: Implications for drug resistance. *Bioinformatics*, 22(24):2975–2979, October 2006.
- [158] K. Deforche, R. Camacho, Z. Grossman, T. Silander, M. A. Soares, Y. Moreau, R. W. Shafer, K. Van Laethem, A. P. Carvalho, B. Wynhoven, P. Cane, J. Snoeck, J. Clarke, S. Sirivichayakul, K. Ariyoshi, A. Holguin, H. Rudich, R. Rodrigues, M. B. Bouzas, P. Cahn, L. F. Brigido, V. Soriano, W. Sugiura, P. Phanuphak, L. Morris, J. Weber, D. Pillay, A. Tanuri, P. R. Harrigan, J. M. Shapiro, D. A. Katzenstein, R. Kantor, and A.-M. Vandamme. Bayesian network analysis of resistance pathways against HIV-1 protease inhibitors. *Infection, Genetics and Evolution*, 7(3):382–390, 2007.
- [159] K. Deforche, R. Camacho, K. Van Laethem, P. Lemey, A. Rambaut, Y. Moreau, and A.-M. Vandamme. Estimation of in vivo fitness landscape experienced by HIV-1 under drug selective pressure useful for prediction of drug resistance evolution during treatment. *Bioinformatics*, 24(1):34–41, 2008.
- [160] A. Liede and S. A. Narod. Hereditary breast and ovarian cancer in Asia: Genetic epidemiology of *BRCA1* and *BRCA2*. *Human Mutation*, 20:413–424, 2002.
- [161] R. H. Lilien, H. Farid, and B. R. Donald. Probabilistic disease classification of expression-dependent proteomic data from mass spectrometry of human serum. *Journal of Computational Biology (JCB)*, 10(6):925–946, 2003.
- [162] R. Fisher. The use of multiple measures in taxonomic problems. *Annals of Eugenics*, 7: 179–188, 1936.
- [163] J. M. Sorace and M. Zhan. A data review and re-assessment of ovarian cancer serum proteomic profiling. *BMC Bioinformatics*, 4(24), June 2003.
- [164] K. A. Baggerly, J. S. Morris, and K. R. Coombes. Reproducibility of SELDI-TOF protein patterns in serum: Comparing data sets from different experiments. *Bioinformatics*, 20(5):777–785, 2004.
- [165] J. Li and L. Wong. Using rules to analyse bio-medical data: A comparison between C4.5 and PCL. In *Proceedings of International Conference on Web-Age Information Management (WAIM)*, pages 254–265, Chengdu, China, 2003. Springer.

REFERENCES

- [166] American Cancer Society. *Cancer Facts & Figures 2007*. American Cancer Society, Atlanta, 2007.
- [167] C. H. Pui and W. E. Evans. Acute lymphoblastic leukemia. *New England Journal of Medicine (NEJM)*, 339:605–615, 1998.
- [168] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, October 1999.
- [169] E.-J. Yeoh, M. E. Ross, S. A. Shurtleff, W. K. Williams, D. Patel, R. Mahfouz, F. G. Behm, S. C. Raimondi, M. V. Relling, A. Patel, C. Cheng, D. Campana, D. Wilkins, X. Zhou, J. Li, H. Liu, C.-H. Pui, W. E. Evans, C. Naeve, L. Wong, and J. R. Downing. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, 1:133–143, March 2002.
- [170] A.-H. Tan and H. Pan. Predictive neural networks for gene expression data analysis. *Neural Networks*, 18:297–306, 2005.
- [171] U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1022–1029. Morgan Kaufmann, 1993.
- [172] J. Li, H. Liu, and L. Wong. Mean-entropy discretized features are effective for classifying high dimensional bio-medical data. In *Proceedings of BIODKDD’03 Workshop on Data Mining in Bioinformatics (with SIGKDD)*, pages 17–24. ACM Press, 2003.
- [173] S. Nikolajewa, R. Pudimat, M. Hiller, M. Platzer, and R. Backofen. BioBayesNet: A web server for feature extraction and Bayesian network modeling of biological sequence data. *Nucleic Acids Research (NAR)*, 35(Web Server):W688–W693, 2007.
- [174] V. Vivet-Boudou, J. Didierjean, C. Isel, and R. Marquet. Nucleoside and nucleotide inhibitors of HIV-1 replication. *Cellular and Molecular Life Sciences (CMLS)*, 63:163–186, 2006.
- [175] N. L. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research (JAIR)*, 5:301–328, 1996.
- [176] G. G. Towell and J. W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13:71–101, 1993.

Appendix A

List of Publications

The following is a list of our publications in internationally refereed journals and conference proceedings. Papers which contain work that has been substantially covered in this thesis are marked with an asterisk (*) at the end.

Journal Papers

- Ghim-Eng Yap, Ah-Hwee Tan, and Hwee-Hwa Pang. Explaining Inferences in Bayesian Networks. *International Journal of Applied Intelligence (APIN)*, Springer Netherlands, Online First, doi: 10.1007/s10489-007-0093-8, October 2007. (*)
- Ghim-Eng Yap, Ah-Hwee Tan, and Hwee-Hwa Pang. Discovering and Exploiting Causal Dependencies for Robust Mobile Context-Aware Recommenders. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 19, no. 7, IEEE Computer Society Press, pp. 977-992, July 2007. (*)

Conference Papers

- Ghim-Eng Yap, Ah-Hwee Tan, and Hwee-Hwa Pang. Learning Causal Models for Noisy Biological Data Mining: An Application to Ovarian Cancer Detection. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI-07)*, Vancouver, British Columbia, AAAI Press, pp. 354-359, July 2007. (*)
- Ghim-Eng Yap, Ah-Hwee Tan, and Hwee-Hwa Pang. Discovering Causal Dependencies in Mobile Context-Aware Recommenders. In *Proceedings of the 7th International Conference on Mobile Data Management (MDM'06)*, Nara, Japan, IEEE Computer Society Press, May 2006. (*)
- Ghim-Eng Yap, Ah-Hwee Tan, and Hwee-Hwa Pang. Dynamically-Optimized Context in Recommender Systems. In *Proceedings of the 6th International Conference on Mobile Data Management (MDM'05)*, Ayia Napa, Cyprus, ACM Press, pp. 265-272, May 2005.

Appendix B

Restaurant Recommender (*RR II*)

B.1 List of Parameters in *RR II*

B.1.1 User-Situational Contexts

- 1.1 Time of visit (24hr-format, hourly from 0700hrs to 2100hrs)
- 1.2 Day of visit (Monday-Sunday)
- 1.3 Period of visit (term time/term break/exam time)
- 1.4 Attire (casual/formal)
- 1.5 Start location (location ID i.e. name of location)
- 1.6 Return location (location ID i.e. name of location)
- 1.7 Maximum duration in hours
- 1.8 Weather (rainy/hot/fine)
- 1.9 Driving (yes/no/maybe)

B.1.2 User-Preferential Contexts

- 2.1 Category (canteen/cafe/club/don't care)
- 2.2 Cleanliness (yes/don't care)
- 2.3 Ventilation (air-con/non air-con/don't care)
- 2.4 Novelty (new/not new/don't care)
- 2.5 Smoking allowed (yes/no/don't care)
- 2.6 Has Toilet (yes/don't care)

CHAPTER B. RESTAURANT RECOMMENDER (RR-II)

- 2.7 Popular with kids (yes/don't care)
- 2.8 Popular with adults (yes/don't care)
- 2.9 Within walking distance (yes/don't care)
- 2.10 Has Chinese vegetarian food (yes/don't care)
- 2.11 Has Chinese cooked food (yes/don't care)
- 2.12 Has Muslim food (yes/don't care)
- 2.13 Has Indian food (yes/don't care)
- 2.14 Has Western food (yes/don't care)
- 2.15 Has Japanese food (yes/don't care)
- 2.16 Has Fast food (yes/don't care)
- 2.17 Cashless payment Available (yes/don't care)

B.1.3 Restaurant Attributes

- 3.1 Operating hours on Monday (as one or more time slot {start-time, end-time})
- 3.2 Operating hours on Tuesday (as one or more time slot {start-time, end-time})
- 3.3 Operating hours on Wednesday (as one or more time slot {start-time, end-time})
- 3.4 Operating hours on Thursday (as one or more time slot {start-time, end-time})
- 3.5 Operating hours on Friday (as one or more time slot {start-time, end-time})
- 3.6 Operating hours on Saturday (as one or more time slot {start-time, end-time})
- 3.7 Operating hours on Sunday (as one or more time slot {start-time, end-time})
- 3.8 Peak hours during term time (as one or more time slot {start-time, end-time})
- 3.9 Peak hours during term break (as one or more time slot {start-time, end-time})
- 3.10 Peak hours during exam time (as one or more time slot {start-time, end-time})
- 3.11 Attire requirement (formal/none)
- 3.12 Has smoking area (yes/no/partial)
- 3.13 Has toilet (yes/no)

CHAPTER B. RESTAURANT RECOMMENDER (RR-II)

- 3.14 Popular with kids (yes/no)
- 3.15 Popular with adults (yes/no)
- 3.16 Renovated within past year (yes/no)
- 3.17 Is clean (yes/no)
- 3.18 Is air-conditioned (yes/no)
- 3.19 Category (canteen/cafe/club)
- 3.20 Easy to park car (yes/no)
- 3.21 Has Chinese vegetarian food (yes/no)
- 3.22 Has Chinese cooked food (yes/no)
- 3.23 Has Muslim food (yes/no)
- 3.24 Has Indian food (yes/no)
- 3.25 Has Western food (yes/no)
- 3.26 Has Japanese food (yes/no)
- 3.27 Has Fast food (yes/no)
- 3.28 Accepts major credit cards (yes/no)
- 3.29 Accepts Electronic Transfer (yes/no)
- 3.30 Location Coordinates (in (x, y)-format)

B.1.4 Restaurant Markers

- 4.1 Opened during visit (yes/no)
- 4.2 Not crowded during visit (yes/no)
- 4.3 User attire is appropriate (yes/no)
- 4.4 Matches “within walking distance” preferences (yes/no)
- 4.5 Matches “smoking allowed” preferences (yes/no)
- 4.6 Matches “has toilet” preferences (yes/no)
- 4.7 Matches “popular with kids” preferences (yes/no)

CHAPTER B. RESTAURANT RECOMMENDER (RR-II)

- 3.14 Popular with kids (yes/no)
- 3.15 Popular with adults (yes/no)
- 3.16 Renovated within past year (yes/no)
- 3.17 Is clean (yes/no)
- 3.18 Is air-conditioned (yes/no)
- 3.19 Category (canteen/cafe/club)
- 3.20 Easy to park car (yes/no)
- 3.21 Has Chinese vegetarian food (yes/no)
- 3.22 Has Chinese cooked food (yes/no)
- 3.23 Has Muslim food (yes/no)
- 3.24 Has Indian food (yes/no)
- 3.25 Has Western food (yes/no)
- 3.26 Has Japanese food (yes/no)
- 3.27 Has Fast food (yes/no)
- 3.28 Accepts major credit cards (yes/no)
- 3.29 Accepts Electronic Transfer (yes/no)
- 3.30 Location Coordinates (in (x, y)-format)

B.1.4 Restaurant Markers

- 4.1 Opened during visit (yes/no)
- 4.2 Not crowded during visit (yes/no)
- 4.3 User attire is appropriate (yes/no)
- 4.4 Matches “within walking distance” preferences (yes/no)
- 4.5 Matches “smoking allowed” preferences (yes/no)
- 4.6 Matches “has toilet” preferences (yes/no)
- 4.7 Matches “popular with kids” preferences (yes/no)

CHAPTER B. RESTAURANT RECOMMENDER (RR_II)

- 4.8 Matches “popular with adults” preferences (yes/no)
- 4.9 Matches “has novelty appeal” preferences (yes/no)
- 4.10 Matches “is clean” preferences (yes/no)
- 4.11 Matches “is air-conditioned” preferences (yes/no)
- 4.12 Matches “is of desired category” preferences (yes/no)
- 4.13 Matches “convenient to drive” preferences (yes/no)
- 4.14 Matches “has Chinese vegetarian food” preferences (yes/no)
- 4.15 Matches “has Chinese cooked food” preferences (yes/no)
- 4.16 Matches “has Muslim food” preferences (yes/no)
- 4.17 Matches “has Indian food” preferences (yes/no)
- 4.18 Matches “has Western food” preferences (yes/no)
- 4.19 Matches “has Japanese food” preferences (yes/no)
- 4.20 Matches “has Fast food” preferences (yes/no)
- 4.21 Matches “cashless payment supported” preferences (yes/no)

B.2 Decision Tree Representations of Marker Rules

Legend

US: userSituations;
 UP: userPreferences;
 RA: restaurantAttributes.

M1. openedDuringVisit

```

US.dayOfVisit = "Monday"
|   operatingHours = RA.operatingHoursOnMonday
US.dayOfVisit = "Tuesday"
|   operatingHours = RA.operatingHoursOnTuesday
US.dayOfVisit = "Wednesday"
|   operatingHours = RA.operatingHoursOnWednesday
US.dayOfVisit = "Thursday"
|   operatingHours = RA.operatingHoursOnThursday
  
```

CHAPTER B. RESTAURANT RECOMMENDER (RR_II)

```

US.dayOfVisit = "Friday"
|   operatingHours = RA.operatingHoursOnFriday
US.dayOfVisit = "Saturday"
|   operatingHours = RA.operatingHoursOnSaturday
US.dayOfVisit = "Sunday"
|   operatingHours = RA.operatingHoursOnSunday
operatingHours = "N.A."
|   openedDuringVisit = no
operatingHours != "N.A."
|   US.timeOfVisit ∈ operatingHours
|   |   openedDuringVisit = yes
|   US.timeOfVisit ∉ operatingHours
|   |   openedDuringVisit = no

```

M2. notCrowdedDuringVisit

```

US.periodOfVisit = "termTime"
|   peakHours = RA.peakHoursDuringTermTime
US.periodOfVisit = "termBreak"
|   peakHours = RA.peakHoursDuringTermBreak
US.periodOfVisit = "examTime"
|   peakHours = RA.peakHoursDuringExamTime
peakHours = "N.A."
|   notCrowdedDuringVisit = yes
peakHours != "N.A."
|   US.timeOfVisit ∈ peakHours
|   |   notCrowdedDuringVisit = no
|   US.timeOfVisit ∉ peakHours
|   |   notCrowdedDuringVisit = yes

```

M3. userAttireIsAppropriate

```

RA.attireRequirement = "formal"
|   US.attire = "formal"
|   |   userAttireIsAppropriate = yes
|   US.attire = "casual"
|   |   userAttireIsAppropriate = no
RA.attireRequirement = "none"
|   userAttireIsAppropriate = yes

```

CHAPTER B. RESTAURANT RECOMMENDER (RR-II)

M4. matchesWithinWalkingDistancePreference

```

UP.withinWalkingDistance = "dunCare"
|   matchesWithinWalkingDistancePreference = yes
UP.withinWalkingDistance = "yes"
|   US.maxDurationInHours > 1.0
|   |   US.weather = "fine"
|   |   |   referenceDistance = [schools.SCE-f&b.C4]
|   |   US.weather = "rainy"
|   |   |   referenceDistance = [schools.SCE-f&b.CA]
|   |   US.weather = "hot"
|   |   |   referenceDistance = [schools.SCE-f&b.CA]
|   US.maxDurationInHours ≤ 1.0
|   |   referenceDistance = [schools.SCE-f&b.CA]
|[US.startLocation-RA.locationCoordinates] > referenceDistance
|   matchesWithinWalkingDistancePreference = no
|[US.startLocation-RA.locationCoordinates] ≤ referenceDistance
|   [US.returnLocation-RA.locationCoordinates] > referenceDistance
|   |   matchesWithinWalkingDistancePreference = no
|   [US.returnLocation-RA.locationCoordinates] ≤ referenceDistance
|   |   matchesWithinWalkingDistancePreference = yes

```

M5. matchesSmokingAllowedPreference

```

UP.smokingAllowed = "yes"
|   RA.hasSmokingArea = "yes"
|   |   matchesSmokingAllowedPreference = yes
|   RA.hasSmokingArea = "no"
|   |   matchesSmokingAllowedPreference = no
|   RA.hasSmokingArea = "partial"
|   |   matchesSmokingAllowedPreference = yes
UP.smokingAllowed = "no"
|   RA.hasSmokingArea = "yes"
|   |   matchesSmokingAllowedPreference = no
|   RA.hasSmokingArea = "no"
|   |   matchesSmokingAllowedPreference = yes
|   RA.hasSmokingArea = "partial"
|   |   matchesSmokingAllowedPreference = yes
UP.smokingAllowed = "dunCare"

```

CHAPTER B. RESTAURANT RECOMMENDER (RR_II)

	RA.hasSmokingArea = "yes"
	matchesSmokingAllowedPreference = yes
	RA.hasSmokingArea = "no"
	matchesSmokingAllowedPreference = yes
	RA.hasSmokingArea = "partial"
	matchesSmokingAllowedPreference = yes

M6. matchesHasToiletPreference

UP.hasToilet = "yes"	
	RA.hasToilet = "yes"
	matchesHasToiletPreference = yes
	RA.hasToilet = "no"
	matchesHasToiletPreference = no
UP.hasToilet = "dunCare"	
	RA.hasToilet = "yes"
	matchesHasToiletPreference = yes
	RA.hasToilet = "no"
	matchesHasToiletPreference = yes

M7. matchesPopularWithKidsPreference

UP.popularWithKids = "yes"	
	RA.popularWithKids = "yes"
	matchesPopularWithKidsPreference = yes
	RA.popularWithKids = "no"
	matchesPopularWithKidsPreference = no
UP.popularWithKids = "dunCare"	
	RA.popularWithKids = "yes"
	matchesPopularWithKidsPreference = yes
	RA.popularWithKids = "no"
	matchesPopularWithKidsPreference = yes

M8. matchesPopularWithAdultsPreference

UP.popularWithAdults = "yes"	
	RA.popularWithAdults = "yes"
	matchesPopularWithAdultsPreference = yes
	RA.popularWithAdults = "no"
	matchesPopularWithAdultsPreference = no

CHAPTER B. RESTAURANT RECOMMENDER (RR.II)

```

UP.popularWithAdults = "dunCare"
|   RA.popularWithAdults = "yes"
|   |   matchesPopularWithAdultsPreference = yes
|   RA.popularWithAdults = "no"
|   |   matchesPopularWithAdultsPreference = yes

```

M9. matchesHasNoveltyAppealPreference

```

UP.novelty = "new"
|   RA.renovatedWithinPastYear = "yes"
|   |   matchesHasNoveltyAppealPreference = yes
|   RA.renovatedWithinPastYear = "no"
|   |   matchesHasNoveltyAppealPreference = no
UP.novelty = "notNew"
|   RA.renovatedWithinPastYear = "yes"
|   |   matchesHasNoveltyAppealPreference = no
|   RA.renovatedWithinPastYear = "no"
|   |   matchesHasNoveltyAppealPreference = yes
UP.novelty = "dunCare"
|   RA.renovatedWithinPastYear = "yes"
|   |   matchesHasNoveltyAppealPreference = yes
|   RA.renovatedWithinPastYear = "no"
|   |   matchesHasNoveltyAppealPreference = yes

```

M10. matchesIsCleanPreference

```

UP.cleanliness = "yes"
|   RA.isClean = "yes"
|   |   matchesIsCleanPreference = yes
|   RA.isClean = "no"
|   |   matchesIsCleanPreference = no
UP.cleanliness = "dunCare"
|   RA.isClean = "yes"
|   |   matchesIsCleanPreference = yes
|   RA.isClean = "no"
|   |   matchesIsCleanPreference = yes

```

M11. matchesIsAirConditionedPreference

```

UP.ventilation = "aircon"

```

CHAPTER B. RESTAURANT RECOMMENDER (RR_II)

```

|      RA.isAirConditioned = "yes"
|      |      matchesIsAirConditionedPreference = yes
|      RA.isAirConditioned = "no"
|      |      matchesIsAirConditionedPreference = no
UP.ventilation = "nonAircon"
|      RA.isAirConditioned = "yes"
|      |      matchesIsAirConditionedPreference = no
|      RA.isAirConditioned = "no"
|      |      matchesIsAirConditionedPreference = yes
UP.ventilation = "dunCare"
|      RA.isAirConditioned = "yes"
|      |      matchesIsAirConditionedPreference = yes
|      RA.isAirConditioned = "no"
|      |      matchesIsAirConditionedPreference = yes

```

M12. matchesIsOfDesiredCategoryPreference

```

UP.category = RA.category
|      matchesIsOfDesiredCategoryPreference = yes
UP.category != RA.category
|      matchesIsOfDesiredCategoryPreference = no

```

M13. matchesConvenientToDrivePreference

```

US.driving = "yes"
|      RA.easyToParkCar = "yes"
|      |      matchesConvenientToDrivePreference = yes
|      RA.easyToParkCar = "no"
|      |      matchesConvenientToDrivePreference = no
US.driving = "no"
|      RA.easyToParkCar = "yes"
|      |      matchesConvenientToDrivePreference = yes
|      RA.easyToParkCar = "no"
|      |      matchesConvenientToDrivePreference = yes
US.driving = "maybe"
|      RA.easyToParkCar = "yes"
|      |      matchesConvenientToDrivePreference = yes
|      RA.easyToParkCar = "no"
|      |      matchesConvenientToDrivePreference = no

```

CHAPTER B. RESTAURANT RECOMMENDER (RR_II)

M14. matchesHasChineseVegetarianFoodPreference

```

UP.hasChineseVegetarianFood = "yes"
|   RA.hasChineseVegetarianFood = "yes"
|   |   matchesHasChineseVegetarianFoodPreference = yes
|   RA.hasChineseVegetarianFood = "no"
|   |   matchesHasChineseVegetarianFoodPreference = no
UP.hasChineseVegetarianFood = "dunCare"
|   RA.hasChineseVegetarianFood = "yes"
|   |   matchesHasChineseVegetarianFoodPreference = yes
|   RA.hasChineseVegetarianFood = "no"
|   |   matchesHasChineseVegetarianFoodPreference = yes

```

M15. matchesHasChineseCookedFoodPreference

```

UP.hasChineseCookedFood = "yes"
|   RA.hasChineseCookedFood = "yes"
|   |   matchesHasChineseCookedFoodPreference = yes
|   RA.hasChineseCookedFood = "no"
|   |   matchesHasChineseCookedFoodPreference = no
UP.hasChineseCookedFood = "dunCare"
|   RA.hasChineseCookedFood = "yes"
|   |   matchesHasChineseCookedFoodPreference = yes
|   RA.hasChineseCookedFood = "no"
|   |   matchesHasChineseCookedFoodPreference = yes

```

M16. matchesHasMuslimFoodPreference

```

UP.hasMuslimFood = "yes"
|   RA.hasMuslimFood = "yes"
|   |   matchesHasMuslimFoodPreference = yes
|   RA.hasMuslimFood = "no"
|   |   matchesHasMuslimFoodPreference = no
UP.hasMuslimFood = "dunCare"
|   RA.hasMuslimFood = "yes"
|   |   matchesHasMuslimFoodPreference = yes
|   RA.hasMuslimFood = "no"
|   |   matchesHasMuslimFoodPreference = yes

```

CHAPTER B. RESTAURANT RECOMMENDER (RR_II)

M17. matchesHasIndianFoodPreference

```

UP.hasIndianFood = "yes"
|   RA.hasIndianFood = "yes"
|   |   matchesHasIndianFoodPreference = yes
|   RA.hasIndianFood = "no"
|   |   matchesHasIndianFoodPreference = no
UP.hasIndianFood = "dunCare"
|   RA.hasIndianFood = "yes"
|   |   matchesHasIndianFoodPreference = yes
|   RA.hasIndianFood = "no"
|   |   matchesHasIndianFoodPreference = yes

```

M18. matchesHasWesternFoodPreference

```

UP.hasWesternFood = "yes"
|   RA.hasWesternFood = "yes"
|   |   matchesHasWesternFoodPreference = yes
|   RA.hasWesternFood = "no"
|   |   matchesHasWesternFoodPreference = no
UP.hasWesternFood = "dunCare"
|   RA.hasWesternFood = "yes"
|   |   matchesHasWesternFoodPreference = yes
|   RA.hasWesternFood = "no"
|   |   matchesHasWesternFoodPreference = yes

```

M19. matchesHasJapaneseFoodPreference

```

UP.hasJapaneseFood = "yes"
|   RA.hasJapaneseFood = "yes"
|   |   matchesHasJapaneseFoodPreference = yes
|   RA.hasJapaneseFood = "no"
|   |   matchesHasJapaneseFoodPreference = no
UP.hasJapaneseFood = "dunCare"
|   RA.hasJapaneseFood = "yes"
|   |   matchesHasJapaneseFoodPreference = yes
|   RA.hasJapaneseFood = "no"
|   |   matchesHasJapaneseFoodPreference = yes

```

M20. matchesHasFastFoodPreference

UP.hasFastFood = "yes"
RA.hasFastFood = "yes"
matchesHasFastFoodPreference = yes
RA.hasFastFood = "no"
matchesHasFastFoodPreference = no
UP.hasFastFood = "dunCare"
RA.hasFastFood = "yes"
matchesHasFastFoodPreference = yes
RA.hasFastFood = "no"
matchesHasFastFoodPreference = yes

M21. matchesCashlessPaymentSupportedPreference

UP.cashlessPaymentAvailable = "yes"
RA.acceptsMajorCreditCards = "yes"
matchesCashlessPaymentSupportedPreference = yes
RA.acceptsMajorCreditCards = "no"
RA.acceptsElectronicTransfer = "yes"
matchesCashlessPaymentSupportedPreference = yes
RA.acceptsElectronicTransfer = "no"
matchesCashlessPaymentSupportedPreference = no
UP.cashlessPaymentAvailable = "dunCare"
RA.acceptsMajorCreditCards = "yes"
matchesCashlessPaymentSupportedPreference = yes
RA.acceptsMajorCreditCards = "no"
RA.acceptsElectronicTransfer = "yes"
matchesCashlessPaymentSupportedPreference = yes
RA.acceptsElectronicTransfer = "no"
matchesCashlessPaymentSupportedPreference = yes

B.3 Decision Tree Representations of Causal Models

Legend

US: userSituations;
UP: userPreferences;
RA: restaurantAttributes.

CHAPTER B. RESTAURANT RECOMMENDER (RR-II)

Causal Model 1

```

UP.category = "cafe"
|   US.attire = "casual"
|   UP.ventilation = "aircon"
UP.category = "club"
|   US.attire = "formal"
|   UP.ventilation = "aircon"
UP.category = "canteen"
|   US.attire = "casual"
|   UP.ventilation = "nonAircon"
UP.category = "dunCare"
|   US.attire = "casual"
|   UP.ventilation = "dunCare"

```

Causal Model 2

```

UP.category = "cafe"
|   UP.hasChineseCookedFood = "dunCare"
|   UP.smokingAllowed = "no"
|   UP.cleanliness = "yes"
UP.category = "club"
|   UP.hasChineseCookedFood = "dunCare"
|   UP.smokingAllowed = "yes"
|   UP.cleanliness = "yes"
UP.category = "canteen"
|   UP.hasChineseCookedFood = "yes"
|   UP.smokingAllowed = "no"
|   UP.cleanliness = "dunCare"
UP.category = "dunCare"
|   UP.hasChineseCookedFood = "dunCare"
|   UP.smokingAllowed = "dunCare"
|   UP.cleanliness = "dunCare"

```

Causal Model 3

```

UP.category = "cafe"
|   UP.hasChineseVegetarianFood = "dunCare"
|   UP.cleanliness = "yes"
UP.category = "club"

```

CHAPTER B. RESTAURANT RECOMMENDER (RR_II)

```

|      UP.hasChineseVegetarianFood = "yes"
|      UP.cleanliness = "yes"
UP.category = "canteen"
|      UP.hasChineseVegetarianFood = "yes"
|      UP.cleanliness = "dunCare"
UP.category = "dunCare"
|      UP.hasChineseVegetarianFood = "dunCare"
|      UP.cleanliness = "dunCare"

```

Causal Model 4

```

UP.category = "cafe"
|      UP.cashlessPaymentAvailable = "yes"
|      UP.smokingAllowed = "no"
|      UP.cleanliness = "yes"
UP.category = "club"
|      UP.cashlessPaymentAvailable = "yes"
|      UP.smokingAllowed = "yes"
|      UP.cleanliness = "yes"
UP.category = "canteen"
|      UP.cashlessPaymentAvailable = "dunCare"
|      UP.smokingAllowed = "no"
|      UP.cleanliness = "dunCare"
UP.category = "dunCare"
|      UP.cashlessPaymentAvailable = "dunCare"
|      UP.smokingAllowed = "dunCare"
|      UP.cleanliness = "dunCare"

```

Causal Model 5

```

UP.category = "cafe"
|      US.attire = "casual"
|      UP.smokingAllowed = "yes"
UP.category = "club"
|      US.attire = "formal"
|      UP.smokingAllowed = "yes"
UP.category = "canteen"
|      US.attire = "casual"
|      UP.smokingAllowed = "no"
UP.category = "dunCare"

```

CHAPTER B. RESTAURANT RECOMMENDER (RR_II)

US.attire = "casual"

UP.smokingAllowed = "dunCare"

Causal Model 6

UP.ventilation = "aircon"

UP.smokingAllowed = "no"

UP.hasChineseVegetarianFood = "dunCare"

UP.smokingAllowed = "yes"

UP.hasChineseVegetarianFood = "yes"

UP.hasIndianFood = "dunCare"

UP.ventilation = "nonAircon"

UP.smokingAllowed = "no"

UP.hasChineseVegetarianFood = "yes"

UP.hasIndianFood = "yes"

UP.ventilation = "dunCare"

UP.smokingAllowed = "dunCare"

UP.hasChineseVegetarianFood = "dunCare"

UP.hasIndianFood = "dunCare"

Causal Model 7

UP.category = "cafe"

UP.cleanliness = "yes"

UP.hasJapaneseFood = "dunCare"

UP.category = "club"

UP.cleanliness = "yes"

UP.hasJapaneseFood = "dunCare"

UP.category = "canteen"

UP.cleanliness = "dunCare"

UP.hasJapaneseFood = "yes"

UP.category = "dunCare"

UP.cleanliness = "dunCare"

UP.hasJapaneseFood = "dunCare"

Causal Model 8

US.driving = "maybe"

US.attire = "casual/formal"

UP.cashlessPaymentAvailable = "yes"

CHAPTER B. RESTAURANT RECOMMENDER (RR-II)

	UP.hasIndianFood = "dunCare"
US.driving = "yes"	
	US.attire = "casual"
	UP.cashlessPaymentAvailable = "dunCare"
	UP.hasIndianFood = "yes"
US.driving = "no"	
	US.attire = "casual"
	UP.cashlessPaymentAvailable = "dunCare"
	UP.hasIndianFood = "dunCare"

Causal Model 9

US.driving = "maybe"	
	UP.hasWesternFood = "dunCare"
	UP.cleanliness = "yes"
US.driving = "yes"	
	UP.hasWesternFood = "yes"
	UP.cleanliness = "dunCare"
US.driving = "no"	
	UP.hasWesternFood = "dunCare"
	UP.cleanliness = "dunCare"

Causal Model 10

UP.category = "cafe"	
	UP.cleanliness = "yes"
	UP.smokingAllowed = "no"
	US.driving = "maybe"
UP.category = "club"	
	UP.cleanliness = "yes"
	UP.smokingAllowed = "yes"
	US.driving = "maybe"
UP.category = "canteen"	
	UP.cleanliness = "dunCare"
	UP.smokingAllowed = "no"
	US.driving = "yes"
UP.category = "dunCare"	
	UP.cleanliness = "dunCare"
	UP.smokingAllowed = "dunCare"
	US.driving = "no"

CHAPTER B. RESTAURANT RECOMMENDER (RR-II)

Causal Model 11

UP.category = "cafe"
 | UP.hasChineseCookedFood = "dunCare"
 | UP.hasWesternFood = "dunCare"
 UP.category = "club"
 | UP.hasChineseCookedFood = "dunCare"
 | UP.hasWesternFood = "dunCare"
 UP.category = "canteen"
 | UP.hasChineseCookedFood = "yes"
 | UP.hasWesternFood = "yes"
 UP.category = "dunCare"
 | UP.hasChineseCookedFood = "dunCare"
 | UP.hasWesternFood = "dunCare"

Causal Model 12

UP.hasChineseVegetarianFood = "yes"
 | UP.hasToilet = "yes"
 | | UP.hasIndianFood = "dunCare"
 | | UP.hasChineseCookedFood = "dunCare"
 | UP.hasToilet = "dunCare"
 | | UP.hasIndianFood = "yes"
 | | UP.hasChineseCookedFood = "yes"
 UP.hasChineseVegetarianFood = "dunCare"
 | UP.hasToilet = "yes"
 | UP.hasIndianFood = "dunCare"
 | UP.hasChineseCookedFood = "dunCare"

Causal Model 13

UP.category = "cafe"
 | US.attire = "casual"
 | UP.hasWesternFood = "dunCare"
 UP.category = "club"
 | US.attire = "formal"
 | UP.hasWesternFood = "dunCare"
 UP.category = "canteen"
 | US.attire = "casual"

CHAPTER B. RESTAURANT RECOMMENDER (RR_II)

| UP.hasWesternFood = “yes”
UP.category = “dunCare”
| US.attire = “casual”
| UP.hasWesternFood = “dunCare”

Causal Model 14

UP.category = “cafe”
| UP.cashlessPaymentAvailable = “yes”
| UP.hasJapaneseFood = “dunCare”
| US.attire = “casual”
UP.category = “club”
| UP.cashlessPaymentAvailable = “yes”
| UP.hasJapaneseFood = “dunCare”
| US.attire = “formal”
UP.category = “canteen”
| UP.cashlessPaymentAvailable = “dunCare”
| UP.hasJapaneseFood = “yes”
| US.attire = “casual”
UP.category = “dunCare”
| UP.cashlessPaymentAvailable = “dunCare”
| UP.hasJapaneseFood = “dunCare”
| US.attire = “casual”

Causal Model 15

US.driving = “maybe”
| UP.ventilation = “aircon”
| UP.hasMuslimFood = “dunCare/yes”
US.driving = “yes”
| UP.ventilation = “nonAircon”
| UP.hasMuslimFood = “yes”
US.driving = “no”
| UP.ventilation = “dunCare”
| UP.hasMuslimFood = “dunCare”

Causal Model 16

UP.category = “cafe”
| UP.hasToilet = “yes”

CHAPTER B. RESTAURANT RECOMMENDER (RR-II)

	UP.hasMuslimFood = "dunCare"
	US.attire = "casual"
UP.category = "club"	
	UP.hasToilet = "yes"
	UP.hasMuslimFood = "yes"
	US.attire = "formal"
UP.category = "canteen"	
	UP.hasToilet = "dunCare"
	UP.hasMuslimFood = "yes"
	US.attire = "casual"
UP.category = "dunCare"	
	UP.hasToilet = "yes"
	UP.hasMuslimFood = "dunCare"
	US.attire = "casual"

Causal Model 17

UP.category = "cafe"	
	UP.hasToilet = "yes"
	UP.hasJapaneseFood = "dunCare"
UP.category = "club"	
	UP.hasToilet = "yes"
	UP.hasJapaneseFood = "dunCare"
UP.category = "canteen"	
	UP.hasToilet = "dunCare"
	UP.hasJapaneseFood = "yes"
UP.category = "dunCare"	
	UP.hasToilet = "yes"
	UP.hasJapaneseFood = "dunCare"

Causal Model 18

UP.category = "cafe"	
	UP.hasToilet = "yes"
	UP.hasIndianFood = "dunCare"
	UP.cleanliness = "yes"
UP.category = "club"	
	UP.hasToilet = "yes"
	UP.hasIndianFood = "dunCare"
	UP.cleanliness = "yes"

CHAPTER B. RESTAURANT RECOMMENDER (RR_II)

```

UP.category = "canteen"
|   UP.hasToilet = "dunCare"
|   UP.hasIndianFood = "yes"
|   UP.cleanliness = "dunCare"
UP.category = "dunCare"
|   UP.hasToilet = "yes"
|   UP.hasIndianFood = "dunCare"
|   UP.cleanliness = "dunCare"

```

Causal Model 19

```

UP.category = "cafe"
|   UP.hasToilet = "yes"
|   US.driving = "maybe"
UP.category = "club"
|   UP.hasToilet = "yes"
|   US.driving = "maybe"
UP.category = "canteen"
|   UP.hasToilet = "dunCare"
|   US.driving = "yes"
UP.category = "dunCare"
|   UP.hasToilet = "yes"
|   US.driving = "no"

```

Causal Model 20

```

UP.category = "cafe"
|   UP.smokingAllowed = "no"
|   UP.hasWesternFood = "dunCare"
|   UP.hasChineseVegetarianFood = "dunCare"
UP.category = "club"
|   UP.smokingAllowed = "yes"
|   UP.hasWesternFood = "dunCare"
|   UP.hasChineseVegetarianFood = "yes"
UP.category = "canteen"
|   UP.smokingAllowed = "no"
|   UP.hasWesternFood = "yes"
|   UP.hasChineseVegetarianFood = "yes"
UP.category = "dunCare"
|   UP.smokingAllowed = "dunCare"

```

CHAPTER B. RESTAURANT RECOMMENDER (RR_II)

UP.hasWesternFood = “dunCare”

UP.hasChineseVegetarianFood = “dunCare”

Causal Model 21

UP.category = “cafe”

| US.attire = “casual”

| UP.smokingAllowed = “no”

| UP.cleanliness = “yes”

| UP.ventilation = “aircon”

| UP.hasChineseVegetarianFood = “dunCare”

| UP.hasChineseCookedFood = “dunCare”

| UP.cashlessPaymentAvailable = “yes”

UP.category = “club”

| US.attire = “formal”

| UP.smokingAllowed = “yes”

| UP.cleanliness = “yes”

| UP.ventilation = “aircon”

| UP.hasChineseVegetarianFood = “dunCare”

| UP.hasChineseCookedFood = “dunCare”

| UP.cashlessPaymentAvailable = “yes”

UP.category = “canteen”

| US.attire = “casual”

| UP.smokingAllowed = “no”

| UP.cleanliness = “dunCare”

| UP.ventilation = “nonAircon”

| UP.hasChineseVegetarianFood = “yes”

| UP.hasChineseCookedFood = “yes”

| UP.cashlessPaymentAvailable = “dunCare”

UP.category = “dunCare”

| US.attire = “casual”

| UP.smokingAllowed = “dunCare”

| UP.cleanliness = “dunCare”

| UP.ventilation = “dunCare”

| UP.hasChineseVegetarianFood = “dunCare”

| UP.hasChineseCookedFood = “dunCare”

| UP.cashlessPaymentAvailable = “dunCare”

CHAPTER B. RESTAURANT RECOMMENDER (RR_II)

Please use the following citation for this dissertation report when referring to any information which is presented in this thesis and not found in our other publications:

Ghim-Eng Yap. *Learning and Exploiting Context Dependencies for Robust Recommendations*. PhD thesis, School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Singapore, 2008.

The corresponding BibTex format for the above citation is given below:

```
@phdthesis {G.-E.Yap:2008,  
  Author = {Ghim-Eng Yap},  
  Title = {Learning and Exploiting Context Dependencies for Robust Recommendations},  
  School = {School of Computer Engineering, Nanyang Technological University},  
  Year = {2008},  
  Address = {Nanyang Avenue, Singapore 639798, Singapore}  
}
```