2014

Analysis of malicious input issues on intelligent systems

Yuanfeng Cai
Iowa State University

Analysis of malicious input issues on intelligent systems

by

Yuanfeng Cai

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Major: Business and Technology (Information Systems)

Program of Study Committee:
Dan Zhu, Major Professor
Zhengrui Jiang
Joey F George
Sree Nilakanta
Huaiqing Wu

Iowa State University
Ames, Iowa
2014

DEDICATION

This dissertation is dedicated to my beloved parents, Yajing Quan and Weihua Cai.

TABLE OF CONTENTS

DEDICATION ii

TABLE OF CONTENTS iii

LIST OF FIGURES vi

LIST OF TABLES viii

ACKNOWLEDGEMENTS ix

ABSTRACT x

CHAPTER 1. INTRODUCTION 1

CHAPTER 2. DESIGNING INTELLIGENT EXPERT SYSTEMS TO COPE WITH LIARS 11

2.1. Introduction 11
2.1.1. Input Distortion 12
2.1.2. Literature Review 13
2.1.3. Contributions 17
2.2. Accuracy-Based Methods 18
2.2.1. A Credit Risk Assessment Example 19
2.2.2. Probability of Being A Liar 21
2.2.3. Split Tree Method 25
2.2.4. Consolidated Tree Method 28
2.3. Value-Based Method 33
2.3.1. Value-Based Split Tree Method 34
2.3.2. Value-Based Consolidated Tree Method 36
2.3.3. Comparison 37
2.4. Experiments for Performance Evaluation 38
2.4.1. Experiment Procedure 38
2.4.1.1. The Basic Procedure. 39
2.4.1.2. The Complete Procedure. 40
2.4.2. Experimental Results 41
2.4.2.1. Liar Percentage. 42
2.4.2.2. Distortion Level. 44
2.4.3. Why TT May Outperform KM? 46
2.4.4. Robustness Tests 49
2.4.4.1. Test on Different VAs. 49
2.4.4.2. Test on Multiple VAs. 49
2.4.4.3. Test on Distortion Matrices. 50
2.4.4.4. Additional Tests based on Simulated Trees. 51
2.4.5. Summary 51

2.5. Selection of VAs 52
2.5.1. Determining the Best VA or VA Group 52
2.5.2. Selecting an External VA 55
2.5.3. Dealing with Strategic Agents 59
2.6. Discussions and Future Research Directions 59

CHAPTER 3. DEVELOP PROFITABLE RECOMMENDATION SYSTEMS WITH SHILLING ATTACK DETECTION 62

3.1. Introduction 62
3.2. Literature Review 65
3.2.1. Collaborative Filtering Recommendation Algorithm 65
3.2.2. Shilling Attack Models 66
3.2.3. Attack Detection 68
3.2.4. Value Based Collaborative Filtering 70
3.3. Value-Based Neighbor Selection 71
3.3.1. Problem Definition 71
3.3.2. The Proposed Method 72
3.3.2.1. Shilling Attacker Probability 73
3.3.2.2. Expected E-retailer's Profit 75
3.3.3. An Illustrative Example 76
3.4. Experimental Results 80
3.4.1. Accuracy-Based Performance Evaluation 81
3.4.2. Value-Based Performance Evaluation 87
3.4.3. Performance Discussion 89
3.5. Discussion and Future Research 91

CHAPTER 4. DESIGNING INTELLIGENT RATING SYSTEMS WITH RATING FRAUD DETECTION 93

4.1. Introduction 93
4.2. Literature Review 96
4.2.1. Rating and Rating Systems 96
4.2.2. Rating Fraud Model 97
4.2.3. Rating Fraud Defense Mechanisms 98
4.3. Rating Fraud Detection 100
4.3.1. Target Entity Rating Series 101
4.3.2. Fraudulent Rater Rating Series 104
4.3.3. Fraudulent Rater Detection Algorithm 106
4.4. Experimental Results 108
4.4.1. Single Attack File Detection Testing 110
4.4.2. Multiple Attack Files Detection Testing 118
4.5. Conclusion 124

CHAPTER 5. CONCLUSIONS AND FUTURE DIRECTIONS 126

5.1 Conclusions 127

5.2 Contribution and Future Research Directions 129

BIBLIOGRAPHY 133

LIST OF FIGURES

Figure 2.1 True Tree for Credit Risk Assessment 21

Figure 2.2 Distribution of Liar and Truth-Teller 23

Figure 2.3(a) Liars' Distortion Matrices for Each Attribute 23

Figure 2.3(b) Truth-Tellers' Distortion Matrices for Each Attribute 23

Figure 2.3(c) The Entire Population's Distortion Matrices for Each Attribute 23

Figure 2.4(a) Liars' Marginal Distributions for Each Attribute 23

Figure 2.4(b) Truth-Tellers' Marginal Distributions for Each Attribute 23

Figure 2.4(c) The Entire Population's Marginal Distributions for Each Attribute 24

Figure 2.5 Split Tree Structure 25

Figure 2.6 Complete Split Tree 28

Figure 2.7 Consolidated Tree 32

Figure 2.8 Misclassification Costs Matrix 34

Figure 2.9 Value-Based Split Tree 36

Figure 2.10 Value-Based Consolidated Tree 37

Figure 2.11 Impact of Liar Percentage on Accuracy 43

Figure 2.12 Impact of Liar Percentage on Misclassification Cost 44

Figure 2.13 Impact of Distortion on Accuracy 45

Figure 2.14 Impact of Distortion on Misclassification Cost 46

Figure 2.15 Mixing Liars and Truth-Tellers 48

Figure 2.16 Multiple Verified Attributes Affect Accuracy 50

Figure 2.17(a) Mis-Recommendation Matrix Corresponding to Bankruptcy 54

Figure 2.17(b) Mis-Recommendation Matrix Corresponding to Education 54

Figure 2.18(a) Distortion Matrices for the External Verified Attribute..... 56

Figure 2.18(b) Marginal Distribution for the External Verified Attribute..... 56

Figure 2.19 Consolidated Tree with External Verified Attribute 57

Figure 2.20 External Verified Attributes Affect Accuracy..... 58

Figure 3.1 The Structure an Attack Profile..... 67

Figure 3.2 Impact of Filler Size and Attack Size on Prediction Error..... 86

Figure 3.3 Impact of Filler Size and Attack Size on Expected Profit..... 86

Figure 4.1 Rating Time Series with Genuine Users 103

Figure 4.2 Rating Time Series with Fraudulent Raters in Various Sizes 103

Figure 4.3 Performance of Target Entity Detection..... 112

Figure 4.4(a) Precision of Fraudulent Raters Detection for the Proposed Method 115

Figure 4.4(b) Recall of Fraudulent Raters Detection for the Proposed Method..... 115

Figure 4.5(a) Precision of Fraudulent Raters Detection for BRS 116

Figure 4.5(b) Recall of Fraudulent Raters Detection for BRS..... 116

Figure 4.6 Rating Shift after Detection in Various Attack Levels..... 117

Figure 4.7(a) Precision of the Proposed Method in Various Attack Sizes 120

Figure 4.7(b) Recall of the Proposed Method in Various Attack Sizes..... 120

Figure 4.8(a) Precision of BRS in Various Attack Sizes..... 121

Figure 4.8(b) Recall of BRS in Various Attack Sizes 121

Figure 4.9 Precision of Fraudulent Rater Detection in Various Intrusion Sizes..... 123

Figure 4.10 Recall of Fraudulent Rater Detection in Various Intrusion Sizes 123

LIST OF TABLES

Table 1.1 A Summary of Malicious Inputs Issues in Three Intelligence Systems 7

Table 1.2 A Summary of s Detection Methods in Three Intelligence Systems 7

Table 2.1 Decision Table for Credit Risk Assessment 20

Table 2.2 Condensed Table for Consolidated Tree Method 32

Table 2.3 Performance Comparison Among Methods 42

Table 3.1 User-Item Rating Matrix 77

Table 3.2 Item Profit Array 77

Table 3.3 Pre-processed information in the database 78

Table 3.4 Comparative Results for MSE 84

Table 3.5 Comparative Results for TEP 85

Table 3.6 Comparative Results between VNS and (SD+HPRS) method 88

Table 4.1 An Example Rating Timestamps Distributions 105

Table 4.2 The Overall Performance for Single Attack File 113

ACKNOWLEDGEMENTS

It is my honor to express my gratitude to my advisor, Dr .Dan Zhu, for her guidance, encouragement, and support over years. I am so thankful to have her as my major professor. She has guided me through the right direction to grow as a researcher. Her inspiriting nature, patience and knowledge have been great source of motivation throughout my studies.

I would also like to thank Dr. Zhengrui Jiang. I was working with him on the lying project for more than three years and have been greatly influenced by his enthusiasm in research and dedication to work. His insightful advices are greatly beneficial for my studies. I am also sincerely grateful to Dr. Joey George, Dr. Sree Nilakanta and Dr. Huaiqing Wu for consenting to serve as my committee members, for their valuable suggestions and feedbacks on my research, and for their continuous support during my doctoral program.

I am fortunate to meet an excellent group of doctoral students in Gerdin, who share their insights and ideas. I would like to especially say thanks to Akshaya Vijayalakshmi, Jenny Lin, Andrew Harrison, Arun Swaminathan and Qazi Kabir. I greatly appreciate for your accompany in this four year journey.

Lastly, but most importantly, thank you, my beloved parents and my boyfriend, Xiao Ma, for your always understanding, support and love.

ABSTRACT

Intelligent systems can facilitate decision making and have been widely applied to various domains. The output of intelligent systems relies on the users' input. However, with the development of Web-Based Interface, users can easily provide dishonest input. Therefore, the accuracy of the generated decision will be affected. This dissertation presents three essays to discuss the defense solutions for malicious input into three types of intelligent systems: expert systems, recommender systems, and rating systems. Different methods are proposed in each domain based on the nature of each problem.

The first essay addresses the input distortion issue in expert systems. It develops four methods to distinguish liars from truth-tellers, and redesign the expert systems to control the impact of input distortion by liars. Experimental results show that the proposed methods could lead to the better accuracy or the lower misclassification cost.

The second essay addresses the shilling attack issue in recommender systems. It proposes an integrated Value-based Neighbor Selection (VNS) approach, which aims to select proper neighbors for recommendation systems that maximize the e-retailer's profit while protecting the system from shilling attacks. Simulations are conducted to demonstrate the effectiveness of the proposed method.

The third essay addresses the rating fraud issue in rating systems. It designs a two-phase procedure for rating fraud detection based on the temporal analysis on the rating series.

Experiments based on the real-world data are utilized to evaluate the effectiveness of the proposed method.

CHAPTER 1. INTRODUCTION

More than ever before, it is apparent that the Internet and Web have changed our lives. Nowadays, more and more customers gather information from the Internet, e.g. product searching or vender comparison. Meanwhile, organizations could also utilize Web applications to retrieve information from their customers' inputs, e.g. order placements or membership applications. Despite the convenience, Web-based interfaces face significant challenges for data processing and decision making. Due to the availability of unprecedented amounts of online data, customers may not be able to retrieve their desired information efficiently while organizations could not process the customers' inputs effectively. Hence, data analytics is used more often by organizations to help process information and make decisions. Accordingly, intelligent systems are becoming increasingly pervasive in organizations.

Intelligent systems are systems with built-in artificial intelligence techniques. In their interaction with users or environment, intelligent systems can exhibit certain aspects of human intelligence including learning, reasoning, memory, adaptability, generalization, flexibility in problematic domains and temporal efficiency (Rudas and Fodor 2008). Hereby, intelligent systems are capable of gathering information, understanding problems, drawing inferences, and generating solutions (Krishnakumar 2003). Given by the input data and embedded algorithms, intelligent systems can be designed to learn and make decisions in various problematic domains such as healthcare, financing, entertainment, and e-commerce.

From the organizational perspective, intelligent systems can facilitate decision making for two types of decision makers: internal employees and external customers. On one hand, organizational employees need to make decisions efficiently given by the tremendous amounts of input data, e.g. healthcare insurance application approval or loan application approval. Intelligent systems can help to make consistent and precise decisions. On the other hand, customers would like to make prior judgments for the risk of their potential future transactions from all of the available information, e.g. which item they may like or which seller may be reliable. Intelligent systems can assist them in finding the most useful information. Hence, applying intelligent systems could provide various advantages to organizations including accurate decision making, reduced decision cycle time and improved customer retention.

Regardless of the type of the decision-maker, the output of the intelligent systems heavily relies on the users' input. For example, a decision on loan applications can be generated based on inputs of customers' personal information; the decision on a preferred item can be made according to inputs of customers' historical purchasing behavior; the decision on seller's reliability can be obtained through inputs of customers' feedback. Thus, the users' input is critical for the systems' decision quality. With the evolution of Web technology, in particular Web 2.0 era, it is more convenient for customers to enter the information since they have access to systems from various locations through Web-based interfaces. This feature has delivered convenience for information collection. However, it brings a challenge to the input data quality. As users are providing information by themselves and it lacks face-to-face interaction, it is difficult to control the truthfulness of their inputs. In

this thesis, we generally term all users who inject dishonest information into the intelligent systems as *malicious users* regardless of their purposes and call their associated information as *malicious inputs*.

The embedded algorithms for various types of intelligent systems are different. Hereby, although the information injected by all malicious users is fraudulent, they behave differently in various systems. In each type of intelligent system, malicious users have the specific objective and so do for malicious input data structure. In this thesis, we focus on three different types of intelligent systems: expert systems, recommender systems and rating systems. Herein, malicious inputs will bring different challenges in each type of systems.

The first challenge is the input distortion issue on intelligent expert systems. Expert systems mimic human experts by applying expertise in a specific domain. Given the users' input, the systems can generate the decision based on the decision rules stored in advance into their knowledge base. However, users may not be willing to disclose their true personal information online except when required to. One possible reason behind this behavior is their concerns about privacy and security. They may falsify input data to protect themselves. Another important factor that contributes to this lying behavior is that self-interested customers may deliberately seek improper benefits by providing incorrect data. For example, during a credit card application, users who are not confident about their financial background may manipulate their partial information so as to get approval. Regardless of the reasons of lying, firms can incur significant costs as a result of incorrect input data. Such type of malicious input is termed as *input distortion* and users who distort their information are

called as *liars*. Input distortion not only decreases the recommendation accuracy, but also increases misclassification cost. In the real-world, misclassification costs are often asymmetric. For instance, classifying a non-worthy customer as a worthy one could potentially be much more costly than classifying a worthy customer as a non-worthy one. Thus, controlling the impact of input distortion from both the accuracy aspect and value aspect is necessary for designing the intelligent expert systems.

The second challenge is the shilling attack issue on intelligent recommender systems. Recommender systems help customers find other users with the similar taste. Different from expert systems, there are no existing decision rules or knowledge bases stored within the systems in advance. Instead, they rely on various similarity- based techniques to find the object to be recommended. One of the most popular techniques adopted in recommender systems is collaborative filtering. For a particular customer (the active user), the collaborative filtering recommender systems will select a list of users who share the similar preference with the active user according to their historical behaviors. The selected users are called as *neighbors*. Based on the neighbors' preference, it can predict the likelihood that a user would be interested in an item that he or she has not seen yet. Hereby, this type of system is vulnerable to malicious inputs from users who deliberately seek improper benefits. They may inject dishonest ratings to promote their own products, which are termed as the *target items*. They provide ratings for non-target items strategically as well so that the similarity between the shilling attackers and other normal users could increase. Accordingly, the target items could be recommended to other honest users more frequently. For the user who introduced biased information so as to lead to the improper recommendations, he or she is termed as

shilling attacker and his or her associated malicious input is termed as *shilling attack*. Different from input distortion behavior, shilling attacker attempts to affect the (recommendation) decision for another user (e.g. the new customer for a particular product), rather than for himself or herself. Even though the active user is willing to disclose the true preference, the recommendation for him or her will still be inaccurate as long as the information gathered from his or her neighbors is incorrect. Shilling attacks will impact the recommendation accuracy for customers, and further influence the customers' satisfaction. In addition, a higher customers' satisfaction is not equal to a higher e-retailer's profit. From the e-retailer's perspective, the ultimate goal is to increase profit (Das et al. 2010). Thus, it is critical to develop intelligent recommender systems to maximize the e-retailer's profit while protecting them from attacks.

The third challenge is the rating fraud issue on intelligent rating systems. Due to the anonymity in the Internet, it may be risky to interact with unfamiliar items or unknown sellers. Rating systems help customers to judge the quality beforehand and reduce the interaction-specific risk. Given by historical feedback from their users, rating systems calculate and disseminate the rating scores for a set of entities. The success of a rating system is determined by the accuracy between the calculated ratings and the true quality of future interactions (Hoffman et al. 2007). However, in order to inflate the reliability of their products or tarnish that of their competitors', some "users" may intentionally input unfairly high or low ratings for a particular set of sellers. Such a type of malicious input is termed as *rating fraud* and the user with such behavior is called as *fraudulent rater*. The rating fraud issue could affect the customers' trust in rating systems, which will further affect their

motivation in participating into the future transactions. Therefore, for organizations relying on customers' online transactions, there is a great need to redesign the intelligent rating systems to defend with rating fraud. While both of the fraudulent raters in the rating systems and the shilling attackers in the recommender systems attack the systems by injecting the malicious ratings, their purposes and their behavior patterns are greatly different. First, an attacker in recommender systems attempts to make certain products more/less visible while a fraudulent rate in ratings systems intend to make certain products more/less reliable. In rating systems, all the products or sellers are visible to their potential customers with the equal chance. Buyers refer to the rating score to decide which product or seller to interact with. Second, in order to make products recommended more frequently, attackers in recommendation systems need to increase the similarity between their rating pattern and other users'. Hereby, shilling attackers have to set up rating profiles to make their rating patterns quite similar to the majority normal users. It means that each shilling attacker should inject at least a certain amount of ratings, which is not the case for fraudulent raters. Actually, the reliability of rating systems may be greatly affected by a group of fraudulent raters each with only a couple of injected ratings, as long as the group size is above a certain level.

Table 1.1 summarizes and compares the challenges induced by malicious inputs in each intelligent system domain. Facing these challenges, this dissertation aims to achieve the following research objectives:

- Develop effective methods against input distortion from both accuracy and value perspectives;

- Develop effective methods against shilling attacks to satisfy customers' needs as well as increase e-retailer's profit;

- Develop effective methods against rating fraud to improve the reliability of rating systems;

Table 1.1 A Summary of Malicious Inputs Issues in Three Intelligence Systems

Chapter	Intelligent Systems Domains	Malicious Inputs	Malicious Users	Malicious Users' Purposes
Chapter 2	Intelligent Expert Systems	Input Distortion	Liars	Get the favorable decision for the liar
Chapter 3	Intelligent Recommender Systems	Shilling Attack	Shilling Attacker	Influence the decision for users other than the shilling attackers
Chapter 4	Intelligent Rating Systems	Rating Fraud	Fraudulent Raters	Influence the decision for users other than the fraudulent raters

Table 1.2 A Summary of Detection Methods in Three Intelligence Systems

Chapter	Malicious Inputs Strategy	Detection Method Proposed	Evaluation Method
Chapter 2	Utilize the existing decision rules to strategically manipulate the input information	ST, CT, VST, VCT	Real-world data and simulation
Chapter 3	Inject ratings to the non-target items strategically to increase the similarity between attackers and the normal users	Value Based Neighbor Selection	Real-world normal data and simulated attack data
Chapter 4	Provide unfair rating collectively to the target entity.	Two Phase Temporal Analysis Method	Real-world normal and attack data

Each of the three dissertation essays aims to achieve one of the objectives stated above with empirical studies in different intelligent systems domains. Table 1.2 summarizes the defense strategies proposed in each dissertation essay. Chapter 2 focuses on the analysis of coping with input distortion from liars in the deductive expert systems. We develop methods to distinguish liars from truth-tellers based on verifiable attributes, and redesign the expert systems to control the impact of input distortion. The four methods we propose are termed *Split Tree* (ST), *Consolidated Tree* (CT), *Value-Based Split Tree* (VST), and *Value-Based Consolidated Tree* (VCT). By comparing the user-provided values and the verified true value for an attribute, it calculates the probability that a user is a liar, and the user is treated differently based on the probability. Among them, ST and CT aim to increase an expert system's accuracy of recommendations, and VST and VCT attempt to reduce the misclassification cost resulting from incorrect recommendations. Experiments are conducted based on both real-world decision tree and simulated trees to compare the performances of the four proposed methods and two existing methods, i.e., the traditional method that ignores input distortion and the knowledge modification (KM) method proposed in a prior research. Results show that the proposed methods can lead to significantly better accuracy or lower cost than existing methods. This result further confirms the advantage of differentiating liars from truth-tellers when such distinctive groups exist in the population.

Chapter 3 focuses on the discussion of dealing with shilling attackers in the collaborative filtering recommender systems. The precision of previous attack detection techniques deteriorates when the attack size or filler size is small, which indicates a larger

probability of misidentifying genuine users as attackers. In addition, seldom do the existing attack detection systems take into consideration the e-retailers' profit when making recommendations. This essay integrates the profitability factor into the traditional systems under the attack environment and proposes an integrated *Value-based Neighbor Selection* (VNS) approach. It selects a proper list of neighbors for recommendation systems that maximize the e-retailer's profit while protecting the system from shilling attacks. The proposed approach is evaluated by a real world dataset against two accuracy-based benchmarks and one value-based benchmark. Results have shown that the proposed approach has realized dual-goals of obtaining recommendation accuracy and sellers' profits. Its advantage is especially significant when either filler size or attack size is small.

Chapter 4 focuses on designing defense mechanisms to detect collaborative rating fraud. In previous literature, the "majority rule" has been adopted as an effective fraud detection technique. By comparing users' ratings of a particular item with its overall mean, it can judge which users are potential fraudulent users according to their rating deviation. However, when the majority of users are fraudulent, the accuracy of this detection technique will decrease. In this essay, it proposes the two-phase detection method based on the analysis of the rating series features. In the first phase, it examines the received rating series of each entity and filter out the entity which is under attack (termed as *target entity*). If the entity is not attacked by the fraudulent raters, its ratings should be random among each other. Otherwise, it is the target entity and the raters for the target entity are the potential fraudulent raters. In the second phase, it analyzes the rating series of each rater selected in the previous phase. Based on its temporal features, a group of fraudulent raters will be discriminated by

using clustering based method. The proposed fraudulent rater detection method is evaluated against a real-world cyber competition data set (Liu et al. 2011). The data set includes both normal data and attack data. Experimental studies have shown that the proposed method is effective in detecting the fraudulent raters accurately while keeping the majority of the normal users in the systems in various attack environment settings.

Chapter 5 concludes the dissertation, summarizing the proposed methods in each essay, stating its contributions to the malicious inputs and intelligent systems' design, discussing its managerial implications in various industry domains, and presenting directions for further studies.

CHAPTER 2. DESIGNING INTELLIGENT EXPERT SYSTEMS TO COPE WITH LIARS

2.1. Introduction

Expert systems are widely applied in diverse fields such as medical treatment, production management, and financial investing (Liao 2005). The mechanism behind expert systems is that they replicate experts' knowledge in specialized domains in the form of decision rules. Given a set of inputs, expert systems produce recommendations to support decision-making. Organization can take advantage of expert systems to reduce the cost of human experts or make better decisions (Duan et al. 2005).

Broadly speaking, there are two types of expert systems: inductive expert systems and deductive expert systems. Inductive expert systems are built using induction algorithms, which develop decisions rules based on pre-classified training datasets (Mookerjee and Dos Santos 1993). Deductive expert systems, on the other hand, are built on deductive algorithms, which derive rules based on existing knowledge base and additional evidence through deductive reasoning (Zhang and Wu 2010). Instead of learning decision rules from training data, decision rules for a deductive expert system could be directly provided by human experts. The focus of this study is on deductive expert systems.

Internet technologies have provided new opportunities for the deployment and wider application of expert systems (Power 2000). Through Web-based interfaces, users can conveniently access an expert system from different locations, and recommendations can be

quickly delivered to them. However, along with the greater convenience, Web-based interfaces also bring real challenges, which we discuss next.

2.1.1. Input Distortion

Input distortion occurs when users do not provide true attribute values to a system. For instance, Hoffman et al. (1999) find that 95% of the users are reluctant to provide information requested by websites. One reason behinds this behavior is the lack of trust between customers and businesses on the Web today (Metzger 2004). Users' concerns about their privacy and information security prevent them from revealing true information. Consequently, users may falsify input data to protect themselves. Another important factor that contributes to this lying behavior is that self-interested customers may deliberately seek improper benefits by providing incorrect data. For example, during a credit card application, users who are not confident about their financial background may manipulate their personal data in order to get approval. This type of lying behavior is further exacerbated by the fact that without face-to-face interaction with users, lying is more difficult and costly to detect.

Regardless of the causes of lying, firms can incur significant costs as a result of incorrect data provided by customers. In the scenario of a credit card application, granting the card to high-risk customers can result in financial losses. On the other hand, incorrectly denying deserving customers can lead to loss of potential revenue and impair firm's reputation.

One intuitive method to deal with the falsified credit application is to impose penalty. Worsham (2010) has reported that prison sentences range from months to years and fines

upwards of $200,000 or more may be charged for falsifying data in a credit application. However, punishments are often costly to enforce, thus may have limited effect on the prevention of input distortion. Another possible method is to utilize incentive mechanisms to discourage users from lying. Incentive mechanisms are easy to carry out but their goals are hardly realized as long as users perceive the benefit of lying is greater than the offered incentive.

Since input distortion is practically impossible to completely eliminate, one may suggest that all user inputs be manually verified to ascertain their accuracy. However, manually verifying user inputs for frequently used expert systems, such as one for consumer credit screening, is typically costly and time-consuming, thus offsetting the benefits of adopting such expert systems. Therefore, manual verification is not a feasible approach to deal with user input distortion for most expert systems. In this study, we focus on automatic approaches to address users' lying to expert systems.

2.1.2. Literature Review

Human's lying behavior has long been studied by researchers. One research stream deals with deception detection. Previous studies suggest that it is possible to use verbal and nonverbal cues to detect deception (Buller and Burgoon 1996; George et al. 2004). In addition, researchers have proposed methods to detect deception via linguistic cues (Zhou et al. 2003; Zhou et al. 2008; Zhou and Zhang 2008). However, these techniques for deception detection cannot be directly applied to address users' lying behavior in our problem context. For example, during an online credit card application, an applicant may only be required to input numeric and simple text information (e.g., name and address). Without face-to-face

contact, it is impossible to capture non-verbal or verbal cues that are critically important for deception detection. Similarly, without rich text information, the linguistic methods cannot be applied. Furthermore, the aforementioned studies on deception detection do not address input distortion for expert systems.

In another stream of research, researchers have developed sophisticated techniques to detect various fraud, such as management fraud (Cecchini et al. 2010), financial fraud (Abbasi et al. 2012), and fake websites (Abbasi et al. 2010). The methods proposed in this stream of research typically build on inductive machine learning algorithms, and require training data that includes a set of fraud cues or contextual information and known classifications. Such techniques cannot be easily applied to detect an individual user's lying behavior because the required fraud clues and contextual information are often difficult to gather from users of deductive expert systems considered in the present research.

The prior literature has also proposes methods to handle noises that can affect the effectiveness of inductive expert systems. For inductive expert systems, noisy data can affect the derived decision rules and subsequently the recommendations. One way of dealing with noisy data is enhancing the quality of training data. Various solutions have been proposed for this purpose, such as class noise identification (Brodley and Friedl 1999: Zhu et al. 2003), erroneous attribute value location (Zhu et al. 2004), and missing attribute value imputation (Fellegi and Holt 1976; Rubin 2004). One problem associated with this type of methods is that important information may be lost during this elimination process (Wu and Zhu 2008). Another important method is decision tree pruning, which could improve the tree

performance under noisy data (Quinlan 1986). Mookerjee et al. (1995) apply the pruning technique during the tree construction phase instead of after it. Boylu et al. (2010) adapt support vector machines (SVM) to generate classifications; their method takes into consideration users' possible strategic behavior such as distorting data. To evaluate their relative performance, Zhou et al. (2004) empirically compare various noise handling techniques and find that only neural networks exhibited consistent performance. Although these solutions improve the performance of inductive experts systems, they all require that a similar error pattern exist in training and testing data, which is not applicable to deductive expert systems.

For deductive expert systems, decision rules are typically provided by domain experts instead of induced from training data, hence we expect that noisy input data affects only the recommendations but not the decision rules used to build an expert system. When formulating decision rules, experts typically assume that the input data at the time of consulting will be accurate. However, as discussed earlier, this is often not the case. To cope with input distortion, Jiang et al. (2005) propose two novel methods to improve the accuracy of recommendations. The first method, termed *Knowledge Modification* (or KM), generates a new decision tree (termed *KM Tree*) based on experts' decision rules as well as users' lying patterns. At the time of consulting, users' input data would be directly fed into the modified decisions tree. The second method, termed *Input Modification*, still uses the decision tree built from decision rules provided by experts, but modifies a user's input data at the time of consulting. Jiang et al. (2005) show that both the Knowledge Modification method and the Input Modification method lead to a significantly improved accuracy than the traditional

method that ignores input distortion, with the Knowledge Modification method outperforming the Input Modification methods under practically all problem scenarios.

Although the KM method proposed by Jiang et al. (2005) substantially increases the accuracy of recommendations under input noises, the method has two limitations. First, the generated KM Tree does not attempt to differentiate liars from truth-tellers at the time of consulting. Instead, all users' inputs are directly fed into the same KM Tree in the same manner. Since the KM method essentially assumes that every user is a liar, it is not effective when there is a clear separation of liars and truth-tellers in the underlying user population. In fact, we find in the present study that when the proportion of truth-tellers is large, the performance of KM method often degrades, sometimes even to a level worse than the accuracy of the traditional method that completely ignores users' lying behavior. Therefore, more intelligent methods that differentiate liars from truth-tellers are warranted under such scenarios. Second, the KM method does not consider misclassification costs when making recommendations. In the real-world, misclassification costs are often asymmetric. For instance, classifying a non-worthy customer as a worthy one could be more costly than classifying a worthy customer as a non-worthy one. The KM method maximizes the expected accuracy of recommendations while completely ignores such misclassification costs. This could lead to suboptimal decisions under some real-world applications. Hence, alternative methods that take into consideration misclassification costs are desirable for certain business applications.

2.1.3. Contributions

To the best of our knowledge, no prior study differentiates liars from truth-tellers and considers misclassification costs when dealing with input noises for deductive expert systems. The present study fills this void and makes two important contributions. Our first major contribution is that we differentiate between liars and truth-tellers in all methods proposed in this study. By comparing the user-provided value and the verified true value for a selected attribute, we calculate the probability that a user is a liar, and the user may be treated differently based on the calculated probability. The first method we propose is termed *Split Tree* (ST). If a customer's probability of liar is above a threshold, she is treated as a liar and her inputs are fed into a *Liar Tree*, which is built based on the pattern of input distortion by liars. Otherwise, she is treated as a truth-teller and the recommendation is generated using the *True Tree* (TT) built directly from decision rules provided by experts. The second method is referred to as *Consolidated Tree* (CT). By taking into consideration all input scenarios under both the True Tree and the Liar Tree, this method generates a new Consolidated Tree. For each possible input vector, the CT method always selects the recommendation with the highest probability of being accurate. We show that both ST and CT lead to better accuracy than the KM method proposed by Jiang et al. (2005).

As the second major contribution, we take into consideration misclassification costs and propose two *value-based* methods that minimize the total misclassification cost resulting from incorrect recommendations. The first value-based method we propose extends the ST method, therefore it is termed *Value-based Split Tree* (VST). The second value-based method is modified based on the CT method and hence is named as *Value-based Consolidated Tree* (VCT). The primary difference between an accuracy-based method (ST or CT) and a value-

based method (VST or VCT) is that the former generates recommendations that maximize accuracy, whereas the latter produces recommendations that minimize the expected misclassification cost. The two value-based methods can be considered generalizations of the accuracy-based methods, and are particular useful when misclassification costs are asymmetric.

The rest of the paper is organized as follows. We develop the two accuracy-based methods in Section 2.2 and the two value-based methods in Section 2.3. In Section 2.4, we report on the experiments conducted for performance evaluation. In Section 2.5, we address some practical issues related to the selection of attributes to determine whether a given user is a liar or not. We conclude the paper in Section 2.6 with discussions on managerial implications and future research directions.

2.2. Accuracy-Based Methods

Deductive expert systems make decisions based on decision rules that are typically provided by human experts. For better efficiency at the time of consulting, such decision rules need to be transformed into a decision tree, which is then used to generate recommendations based on inputs provided by users. To differentiate it from decision trees generated from other methods, we refer to the decision tree built directly from expert-provided decision rules as the *True Tree* (*TT*).

When the decision rules are formulated, experts implicitly assume that all attribute values provided by users are accurate. For instance, an expert may classify a customer who

claims to have medium income and full-time employment as low-risk. An implicit assumption behind this decision rule is that the customer indeed has a medium income and a full-time job. However, as discussed earlier, users may lie when providing inputs to an expert system. In the same example, if a customer who claims to have a medium income and a full-time job is actually unemployed with no significant income, then classifying the customer as a low-risk one can lead to financial losses. As previously mentioned, facing such lying behavior, a *KM Tree*, built based on the KM method proposed by Jiang et al. (2005), can replace the True Tree to serve as the "expert" at the time of consulting. The KM method, however, does not differentiate liars from truth-tellers, and hence leaves room for improvement.

In this section, we propose two accuracy-based methods that explicitly separate liars from truth-tellers: *Split Tree* (ST) and *Consolidated Tree* (CT). In constructing their own decision trees, both methods first estimate the probability that a particular customer is a liar, and then generate recommendations based on the calculated probability and a set of parameter values for the underlying user population.

To illustrate the typical problem context for deductive expert systems and the different noise handling methods, we first present a credit risk assessment example that is similar to the one used by Jiang et al. (2005).

2.2.1. A Credit Risk Assessment Example

In order to decide whether to provide credit to potential customers, firms need to first assess their credit-worthiness. Human experts could provide a set of decision rules, as

illustrated in Table 2.1, that can be used for such an assessment. Based on rules represented in this table, a customer can be classified into three risk levels, i.e., low risk (LR), medium risk (MR) and high risk (HR), based on the values of four attributes: *Income* (high, medium, low), *Bachelor's Degree* (yes, no), *Employment* (yes, no), and *Bankruptcy* (yes, no). The dash entry ("-") in the table means that the value for that attribute "does not matter" for a given decision rule. For instance, rule R0 classifies a customer as low-risk as long as the customer's income is high, regardless of whether the customer has a degree, a bankruptcy record, or a job. Since the decision rules shown in Table 2.1 are provided based on the assumption that all attribute values are correct, they represent the *True Table*. Based on heuristic algorithms, the True Table can be translated into a True Tree, as shown in Figure 2.1. At the time of consulting, the True Tree, instead of the True Table, should be used because the former is more efficient than the latter.

Table 2.1 Decision Table for Credit Risk Assessment

Attributes / Rules	Income	Bachelor's Degree	Employment	Bankruptcy	Classification
R0	H	-	-	-	**LR**
R1	M	Y	Y	-	**LR**
R2	M	-	N	-	**MR**
R3	M	N	Y	-	**MR**
R4	L	-	-	Y	**HR**
R5	L	-	-	N	**MR**

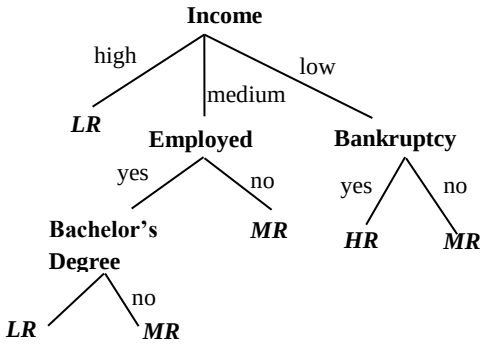

Figure 2.1 True Tree for Credit Risk Assessment

2.2.2. Probability of Being A Liar

In order to estimate the probability that a particular user is a liar, we need to compare the value provided by the user and the corresponding true value for at least one attribute. The attribute used for the verification purpose should be relatively easy to validate. We call such an attribute *Verifiable Attribute* (or VA). In the credit risk assessment example, *Bankruptcy* could be used as a VA since it can be relatively quickly and cost-efficiently obtained from a customer's credit report. Given both the observed and true values of a VA, denoted by VA^O and VA^T respectively, the conditional probability that the user is a liar can be derived based on the Bayes' Theorem:

$$\mathrm{P}(\text{liar} \mid VA^O, VA^T) = \frac{\mathrm{P}(VA^O, VA^T \mid \text{liar}) \cdot \mathrm{P}(\text{liar})}{\mathrm{P}(VA^O, VA^T \mid \text{liar}) \cdot \mathrm{P}(\text{liar}) + \mathrm{P}(VA^O, VA^T \mid \text{truth-teller}) \cdot \mathrm{P}(\text{truth-teller})}, \quad (1)$$

where

$$P(VA^O, VA^T \mid \text{liar}) = P(VA^O \mid VA^T, \text{liar}) \cdot P(VA^T, \text{liar}),$$

$$\mathrm{P}(VA^O, VA^T \mid \text{truth-teller}) = \mathrm{P}(VA^O \mid VA^T, \text{truth-teller}) \cdot \mathrm{P}(VA^T, \text{truth-teller}).$$

Given (1), we have

$$\mathrm{P}(\text{truth-teller} \mid VA^O, VA^T) = 1 - \mathrm{P}(\text{liar} \mid VA^O, VA^T). \quad (2)$$

The conditional and marginal probabilities in (1) can be obtained from historical data or through sampling. During the sampling process, a certain number of users are selected and their true attribute values verified. A user who lied about at least one attribute is classified as a liar; those who did not lie about any attribute are classified as truth-tellers. Based on this classification, we can estimate the distribution of liars and truth-tellers in the population, as illustrated in Figure 2.2. In addition, we can estimate the *distortion matrixes* for liars and truth-tellers for each attribute, as shown in Figures 2.3(a) and 2.3(b). We then calculate the distortion matrixes for the entire user population (including both truth-tellers and liars) for all attributes, as shown in Figure 2.3(c). In a distortion matrix, the rows represent the true values, the columns record the observed values, and the numbers captures the conditional probability of every observed attribute value given each true value. For instance, the first 0.3 in the liar's distortion matrix for *Income* implies that among those whose true income is low, 30% are likely to claim that their income is high. Through sampling, we can also estimate the marginal distribution of true attribute values for both liars and truth-tellers, as shown in Figures 2.4(a) and 2.4(b). Here, the marginal distributions need to be separately estimated for liars and truth-tellers. The marginal distributions for the entire user population that includes both truth-teller and liars are calculated and shown as in Figure 2.4(c). Note that these distortion matrices and marginal distributions are similar to those estimated by Jiang et al. (2005). However, Jiang et al. (2005) do not estimate them separately for liars and truth-tellers. Instead, distortion matrices and marginal distributions are estimated based on data for all users in the sample, essentially the same as those shown in Figures 2.3(c) and 2.4(c).

Liar (L)	0.4
Truth-Teller (T)	0.6

Figure 2.2 Distribution of Liar and Truth-Teller

Income (I)

	High	Medium	Low
High	0.85	0.075	0.075
Medium	0.225	0.55	0.225
Low	0.3	0.3	0.4

Bachelor's Degree (D)

	Yes	No
Yes	0.9	0.1
No	0.4	0.6

Employed (E)

	Yes	No
Yes	0.85	0.15
No	0.90	0.10

Bankruptcy (B)

	Yes	No
Yes	0.11	0.89
No	0.01	0.99

Figure 2.3(a) Liars' Distortion Matrices for Each Attribute

Income (I)

	High	Medium	Low
High	1.0	0.0	0.0
Medium	0.0	1.0	0.0
Low	0.0	0.0	1.0

Bachelor's Degree (D)

	Yes	No
Yes	1.0	0.0
No	0.0	1.0

Employed (E)

	Yes	No
Yes	1.0	0.0
No	0.0	1.0

Bankruptcy (B)

	Yes	No
Yes	1.0	0.0
No	0.0	1.0

Figure 2.3(b) Truth-Tellers' Distortion Matrices for Each Attribute

Income (I)

	High	Medium	Low
High	0.95	0.025	0.025
Medium	0.09	0.82	0.09
Low	0.245	0.245	0.51

Bachelor's Degree (D)

	Yes	No
Yes	0.97	0.03
No	0.2	0.8

Employed (E)

	Yes	No
Yes	0.94	0.06
No	0.42	0.58

Bankruptcy (B)

	Yes	No
Yes	0.2	0.8
No	0.002	0.998

Figure 2.3(c) The Entire Population's Distortion Matrices for Each Attribute

***Income* (*I*)**

High	0.5
Medium	0.3
Low	0.2

***Bachelor's Degree* (*D*)**

Yes	0.55
No	0.45

***Employed* (*E*)**

Yes	0.65
No	0.35

***Bankruptcy* (*B*)**

Yes	0.45
No	0.55

Figure 2.4(a) Liars' Marginal Distributions for Each Attribute

***Income* (*I*)**

High	0.67
Medium	0.3
Low	0.03

***Bachelor's Degree* (*D*)**

Yes	0.7
No	0.3

***Employed* (*E*)**

Yes	0.73
No	0.27

***Bankruptcy* (*B*)**

Yes	0.03
No	0.97

Figure 2.4(b) Truth-Tellers' Marginal Distributions for Each Attribute

***Income* (*I*)**

High	0.6
Medium	0.3
Low	0.1

***Bachelor's Degree* (*D*)**

Yes	0.64
No	0.36

***Employed* (*E*)**

Yes	0.7
No	0.3

***Bankruptcy* (*B*)**

Yes	0.2
No	0.8

Figure 2.4(c) The Entire Population's Marginal Distributions for Each Attribute

Given the distributions and conditional probabilities included in the distortion matrices, we can estimate the probability that a user is a liar given the true and observed values of an attribute. To illustrate, consider an observed vector (I^O = "H", D^O = "N", E^O = "N", B^O = "N"), representing the observed values of *Income*, *Bachelor's Degree*, *Employment*, and *Bankruptcy*, and a verified true *VA* (Bankruptcy) value B^T = "N". We next show how to use the numbers shown in Figures 2.2-2.4 to estimate the conditional probability that this customer is a liar. Based on (1), we first obtain

$$P(B^O = \text{"N"}, B^T = \text{"N"} \mid \text{liar}) = P(B^O = \text{"N"} \mid B^T = \text{"N"}, \text{liar}) \cdot P(B^T = \text{"N"}, \text{liar}) = 0.99 * 0.55, \text{ and}$$

$$\begin{aligned} &\mathrm{P}(B^O = \text{"N"}, B^T = \text{"N"} \mid \text{truth–teller}) \\ &\quad = \mathrm{P}(B^O = \text{"N"} \mid B^T = \text{"N"}, \text{truth-teller}) \cdot \mathrm{P}(B^T = \text{"N"}, \text{truth-teller}) = 1.0 * 0.97. \end{aligned}$$

Then,

$$\begin{aligned} &\mathrm{P}(\text{liar} \mid B^O = \text{"N"}, B^T = \text{"N"}) \\ &= \frac{\mathrm{P}(B^O = \text{"N"}, B^T = \text{"N"} \mid \text{liar}) \cdot \mathrm{P}(\text{liar})}{\mathrm{P}(B^O = \text{"N"}, B^T = \text{"N"} \mid \text{liar}) \cdot \mathrm{P}(\text{liar}) + \mathrm{P}(B^O = \text{"N"}, B^T = \text{"N"} \mid \text{truth-teller}) \cdot \mathrm{P}(\text{truth-teller})} \\ &= \frac{0.99 * 0.55 * 0.4}{0.99 * 0.55 * 0.4 + 1.0 * 0.97 * 0.6} = 0.272. \end{aligned}$$

The result shows that even if the observed and true values of Bankruptcy are both "N," the user still has a 27.2% chance of being a liar. This has an important implication. Even though the observed value is the same as the true one, it does not guarantee that the customer

is a truth-teller. On the other hand, a user is a liar with certainty if the observed value is not the same as the truth value. This can be verified based on (1). If B^O is not equal to B^T, P(B^O | B^T, truth-teller) = 0. Thus, P(liar| $B^O \neq B^T$) = 1.

2.2.3. Split Tree Method (ST)

Once the probability that a given user is a liar is estimated, we can decide whether to use the True Tree or a tree specifically built for liars, named *Liar Tree*, to generate recommendations. As shown in Figure 2.5, if the Threshold is set at 0.5, then the True Tree should be consulted if the probability that a user being a liar is 0.272, as calculated in the preceding example. On the other hand, the Liar Tree is consulted if the user's probability of being a liar is higher than 0.5. We call this method a *Split Tree* (or ST) method. Under the ST method, the True Tree is directly constructed from the expert-provided decision rules; the Liar Tree can be built following the same steps for building a KM Tree (Jiang et al. 2005), with the exception that Liar's distortion matrices and marginal distributions, instead of those for all users, are used. We next briefly describe the steps used to generate the Liar Tree.

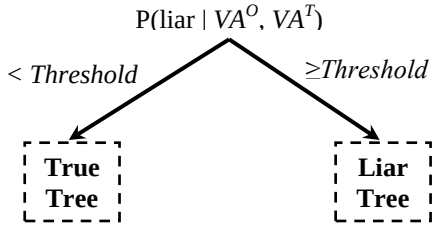

Figure 2.5 Split Tree Structure

To build the Liar Tree, we first need to calculate the *Liar's decision table* (or *Liar table*), which includes recommendations for all possible observed vectors. For instance, for

the credit risk assessment example, there are $3 \times 2 \times 2 \times 2 = 24$ rules in the Liar table. Similarly, if there are 10 binary attributes, the fully enumerated Liar table will include 2^{10} rules. The recommendation for each possible observed vector is computed using the following steps:

Step 1: Given an observed vector, calculate the probabilities associated with every possible true vector.

Denote the observed and a true vector by **Observed** and **True**, respectively. The conditional probability is

$$\mathrm{P}(\boldsymbol{True} \mid \boldsymbol{Observed}) = \frac{\mathrm{P}(\boldsymbol{Observed} \mid \boldsymbol{True}) \cdot \mathrm{P}(\boldsymbol{True})}{\mathrm{P}(\boldsymbol{Observed})}. \quad (3)$$

where r is the index over all true vectors. If the number of possible true vectors is large, we can assume

$$\mathrm{P}(\boldsymbol{True}) = \Pi_i \mathrm{P}(True_i),$$

where $True_i$ represents the value for attribute i in the ***True*** vector. Similar to the well-known naïve Bayes method, we adopt two other assumptions:

$$\mathrm{P}(\boldsymbol{Observed} \mid \boldsymbol{True}) = \Pi_i \mathrm{P}(Observed_i \mid \boldsymbol{True}) \text{ for all } i, \text{ and}$$

$$\mathrm{P}(Observed_i \mid \boldsymbol{True}) = \mathrm{P}(Observed_i \mid True_i) \text{ for all } i.$$

Then,

$$\mathrm{P}(\boldsymbol{Observed} \mid \boldsymbol{True}) = \Pi_i \mathrm{P}(Observed_i \mid True_i). \quad (4)$$

Finally, $\mathrm{P}(\boldsymbol{Observed})$ can be calculated based on the law of total probability, i.e.,

$$\mathrm{P}(\boldsymbol{Observed}) = \sum_r \mathrm{P}(\boldsymbol{Observed} \mid \boldsymbol{True}^r)\mathrm{P}(\boldsymbol{True}^r), \quad (5)$$

where r is the index over all possible true vectors.

Note that in calculating the conditional probabilities, we need to use the distortion matrices and marginal distributions for liars, as illustrated in Figures 2.3(a) and 2.4(a).

Step 2: Find the Liar Table recommendation.

Use the True Tree to obtain the recommendation for every true vector. Add the conditional probabilities associated with all true vectors that have the same recommendation. The recommendation with the highest total probability is selected as the Liar Table recommendation for the **Observed** vector, because this recommendation is most likely to be the accurate one for a user with the **Observed** vector.

Step 3: Generate and Condense the Liar Table.

Repeat Step 1 and Step 2 for all possible observed vectors to generate the fully enumerated Liar Table. In the full Liar Table, sometimes the value of a given attribute does not affect the recommendation. Therefore, we can collapse each set of "redundant" decision rules into a single row, similar to those shown in Table 2.1. Once all such redundancies are removed, we obtain a condensed Liar Table.

Step 4: Build the Liar Tree from the condensed Liar Table.

The heuristic used to build the True Tree can be used to construct the Liar Tree (or LT for short).

Following steps 1-4, and using data shown in Figures 2.2-2.4, we obtain a Liar Tree for the credit risk assessment example. The complete Split Tree is shown in Figure 2.6. For better clarity, the True Tree structure is not included in the figure.

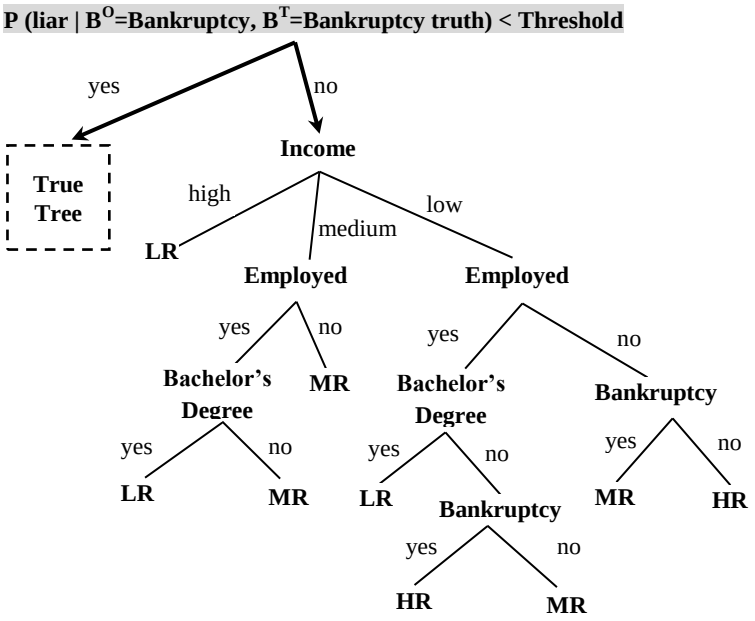

Figure 2.6 Complete Split Tree

2.2.4. Consolidated Tree Method (CT)

With the ST method, the VA is always verified before deciding which tree branch to traverse. After additional analysis, we find that under certain situations, the true value of the VA does not affect the recommendation. Motivated by this observation, we develop an alternative method to deal with users' input distortion. With this new method, the true value of the VA is simply treated as a "separate" attribute in the decision table. Therefore, each vector in the expanded decision includes the observed values of all attributes as well as the true value of the VA. The best recommendation for this vector is one with the highest probability of being correct given the available information. The expanded decision table (or

the CT Table) is then condensed and transformed into a single tree. We call this method *Consolidated Tree* (or CT) method because there are no separate tree branches for liars and truth-tellers under this method. We next explain in detail how the Consolidated Tree can be constructed.

For each vector in the CT Table, repeat Steps 1-4 to obtain its CT recommendation:

Step 1. Find the probability that the user with the given vector is a liar.

Since the given vector includes both the observed value and the true value of the VA, the probability that the user is a liar or a truth-teller can be calculated based on formulas (1) and (2). We denote these probabilities by P(liar | VA^O, VA^T) and P(truth-teller | VA^O, VA^T), respectively.

Step 2. Calculate LT path probability associated with each possible recommendation.

Since there is one Consolidated Tree, we need to consider the probability that a user is liar as well as the probability that she is a truth-teller. If the user is a liar, then similar to Step 1 in building the Liar Tree, we calculate the conditional probabilities associated with all possible true vectors and then sum up the conditional probabilities of all true vectors that have the same recommendation. Using the credit risk assessment example, we denote the total conditional probabilities associated with low-risk, medium risk, and high-risk by P(LR), P(MR), and P(HR), respectively. Since these probabilities are relevant only if the user is a liar, we need to multiply each of them by P(liar| VA^O, VA^T) and obtain P(liar | VA^O, VA^T)*P(LR) , P(liar | VA^O, VA^T)*P(MR), and P(liar | VA^O, VA^T)*P(HR). Each of these probabilities is referred to as *LT path probability* associated with a recommendation.

Step 3. Calculate the TT path probability.

If P(truth-teller | VA^O, VA^T) > 0, feed the observed values into the True Tree, and obtain the *True recommendation*. Since there is no uncertainty in the Truth Tree path, we set the *TT path probability* associated with the True recommendation as P(truth-teller | VA^O, VA^T) and that associated with all other recommendations as zero.

Step 4. Obtain the CT recommendation for the given vector.

Add the LT path probabilities and the TT path probabilities for the same recommendations. The recommendation with the highest probability sum is selected as the CT recommendation for the given vector. For instance, if the True recommendation is MR for the credit risk assessment example, then we need to compare P(liar | VA^O, VA^T)* P(MR) + P(truth-teller | VA^O, VA^T) with P(liar | VA^O, VA^T) * P(LR) and P(liar | VA^O, VA^T) * P(HR). The recommendation with the highest probability is selected as the *CT* recommendation.

Step 5: Repeat Steps 1 – 4 for all possible vectors to generate the fully enumerated CT Table.

Step 6: Condense the CT Table and transform it into a CT Tree.

Following the CT method, the condensed table for the credit assessment example is shown in Table 2.2. Note that "*Bankruptcy Truth*" is treated as the fifth attribute in the table. The "-" symbol in the fifth column implies that whether the true *Bankruptcy* value is true or

not does not affect the recommendations produced by the CT method, hence the attribute need not be verified. This is an important advantage of the CT method over the ST method.

From the CT Table, we obtain the corresponding Consolidated Tree, as shown in Figure 2.7. By comparing the Split Tree shown in Figure 2.6 and the Consolidated Tree shown in this figure, we find that these two methods do not always lead to the same recommendation. Furthermore, this Consolidated Tree shows that a users' true *Bankruptcy* value does not need to be verified in every branch. This can lead to significant cost savings because verifying the true values of a VA takes time and incurs costs. Therefore, the CT method is more efficient than the ST method. We also expect the CT method to have a better accuracy than the ST method because when the CT Tree is built, we maximize the accuracy of recommendation by taking into consideration all possible paths and their associated probabilities.

Table 2.2 Condensed Table for Consolidated Tree Method

Income	Bachelor's Degree	Employment	Bankruptcy	Bankruptcy Truth	Classification
H	-	-	-	-	**LR**
M	Y	Y	-	-	**LR**
M	N	Y	-	-	**MR**
M	-	N	-	-	**MR**
L	-	-	Y	Y	**HR**
L	-	N	Y	N	**HR**
L	Y	Y	Y	N	**MR**
L	N	Y	Y	N	**HR**
L	Y	-	N	N	**MR**
L	Y	Y	N	Y	**LR**
L	Y	N	N	Y	**MR**
L	N	Y	N	-	**MR**
L	N	N	N	Y	**MR**
L	N	N	N	N	**LR**

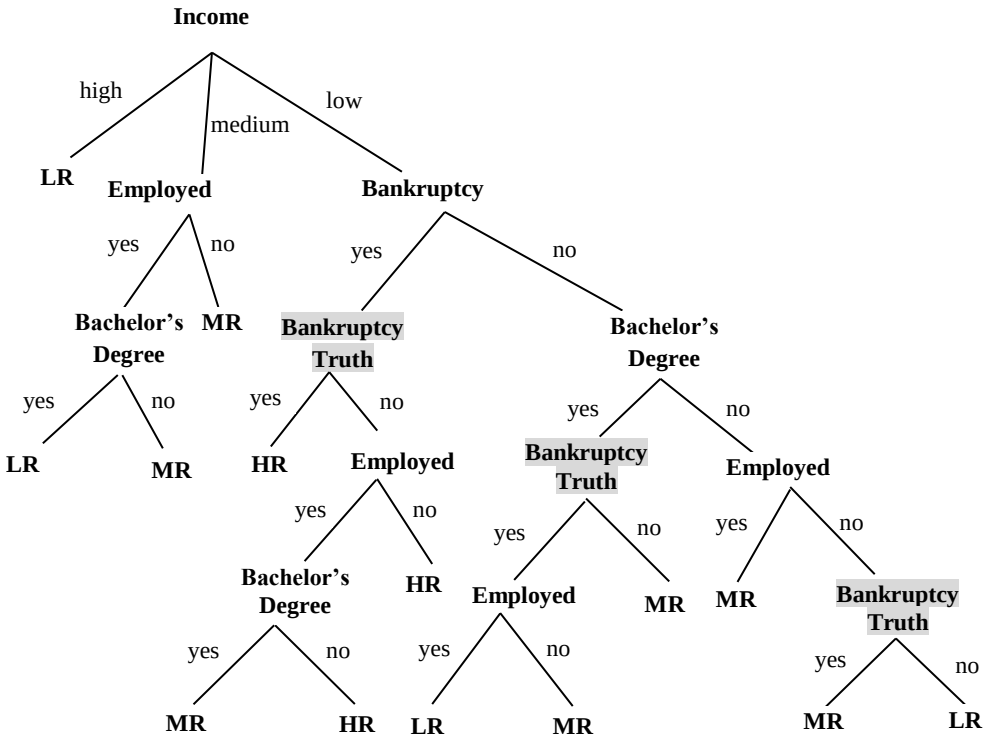

Figure 2.7 Consolidated Tree

2.3. Value-Based Method

The ST and CT methods proposed in Section 2.2 attempt to maximize the accuracy of recommendations. An implicitly assumption made in the two methods is that the misclassification cost remains the same for all misclassification scenarios. In the real-world, this may not be the case. For instance, incorrectly classifying a low-risk customer as a high risk one may lead to the denial of loan and hence the loss of opportunity to earn interests from the customer. On the other hand, misclassifying a high risk customer as a low-risk one may lead to default. The second scenario can potentially be much more costly than the first scenario. In this section, we extend the two methods developed in the previous section to the corresponding value-based methods: *Value-based Split Tree* (VST) and *Value-based Consolidated Tree* (VCT). The main difference between the accuracy-based methods (ST and CT) and the value-based methods (VST and VCT) is that the former attempts to maximize the accuracy of recommendations, while the later makes recommendations that lead to the lowest misclassification costs.

In order to use the value-based methods, we need to know the misclassification costs under different scenarios. For convenience of exposition, we first construct the *misclassification cost matrix*. In a misclassification cost matrix, the columns represent the true class, and the rows represent the recommended class. Figure 2.8 shows a hypothetical matrix for the credit risk assessment example. In this matrix, the entry "20" means that the cost of misclassifying a low-risk customer as a high-risk one is 20. In the real world, misclassification costs can be estimated by domain experts or from historical data. For

instance, financial institutions often collect data regarding loan default rates from customers at different risk levels.[1] Such data could be used to construct the misclassification cost matrix for a credit risk assessment application.

	LR	MR	HR
LR	0	45	100
MR	10	0	50
HR	20	28	0

Figure 2.8 Misclassification Costs Matrix

2.3.1. Value-Based Split Tree Method (VST)

The input of VST is similar to that of ST method, except that VST takes into consideration the misclassification costs when deciding the best recommendations. Specifically, the method traverses the True Tree branch recommendation when the probability of liar is below a threshold. Otherwise, it traverses the Value-based Liar Tree branch. The difference between Value-based Liar Tree and Liar Tree is that the former is constructed based on recommendations that minimize the misclassification costs instead of maximizing the probability of being accurate.

In terms of the actual procedure, ST and VST differ only in Step 2 that generates the (Value-based) Liar Table recommendations. In Step 2 of the ST method, we add the conditional probabilities associated with the same recommendations and select the recommendation with the highest probability sum. In Step 2 of the VST method, we first

[1] An example of such data is available at http://en.wikipedia.org/wiki/Credit_card_interest.

obtain the same probability sums for all possible recommendations, then calculate the misclassification costs under different scenarios, and finally select the recommendation that minimizes the expected total misclassification cost.

Formally, let *GR* denote the generated recommendation and *AR* the accurate recommendation. The recommendation with the lowest misclassification cost can be obtained by

$$\underset{GR^q}{\operatorname{argmin}}\ C(GR^q) = \sum_l C(GR^q \mid AR^l) \cdot \mathrm{P}(AR^l), \tag{5}$$

where l is the index over all possible accurate recommendations, q is the index over all possible generated recommendations, $\mathrm{P}(AR^l)$ is the same probability sum associated with AR^l obtained in Step 2 of the ST method, and $C(GR^q|AR^l)$ is the misclassification cost and can be found from the misclassification cost matrix.

Figure 2.9 shows the Value-based Split Tree built for the credit risk assessment example. Similar to the ST method, the VST method produces a tree structure that includes a True Tree branch and a Valued-based Liar Tree branch. Once again, the branch traversed at the time of consulting depends on whether the probability that a user is a liar is above or below the given threshold.

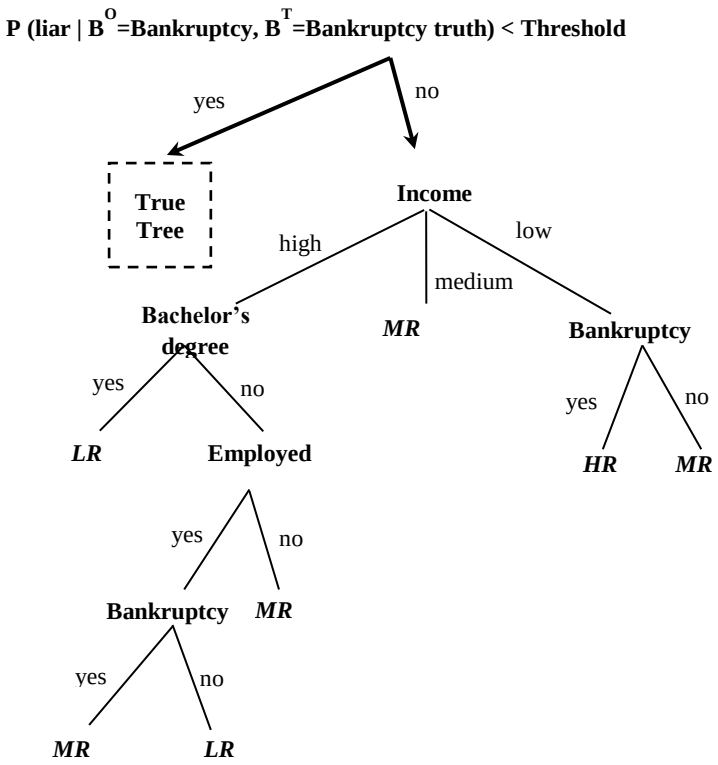

Figure 2.9 Value-Based Split Tree

2.3.2. Value-Based Consolidated Tree Method (VCT)

Analogous to the extension from ST to VST, we now extend the accuracy-based CT method to produce a *Value-based Consolidated Tree* (VCT) that minimizes the expected total misclassification cost.

Similar to the difference between ST to VST, the procedures for CT and VCT differ only in the step that selects the CT (VCT) recommendations. Specifically, in Step 4 of the CT method, we add LT and TT path probabilities for every recommendation and then select the recommendation with the highest probability sum as the CT recommendation. In Step 4 of the VCT method, we still calculate the sum of the LT and TT path probabilities associated

with every recommendation. Once the probability sums are obtained, we can use again use formula (5) to obtain the expected total misclassification cost $C(GR^q)$ associated with each recommendation GR^q, and then select the recommendation with the lowest total misclassification cost as the VCT recommendation.

The final output of VCT method is Value-based Consolidated Tree. Figure 2.10 illustrates such a tree for the credit risk assessment example.

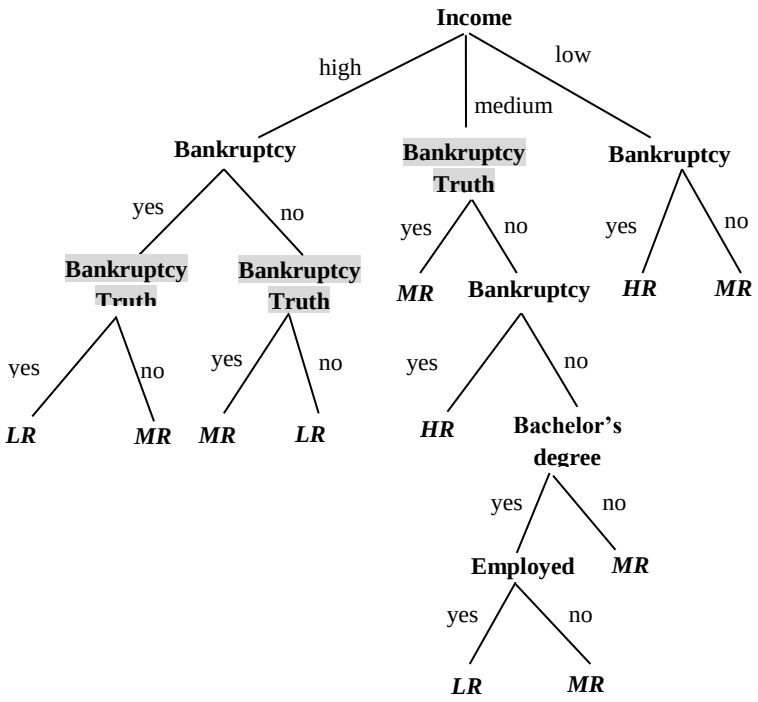

Figure 2.10 Value-Based Consolidated Tree

2.3.3. Comparison

As explained earlier, the Split Tree methods always require the verification of the true value of the VA, whereas the Consolidated Tree methods may need to verify VA only for

certain cases. The VST and VCT trees shown in Figures 2.9 and 2.10 again confirm this important difference.

Furthermore, the differences between Figures 2.6 and 2.9 or those between Figures 2.7 and 2.10 show that the most accurate recommendation is not always the one that minimizes the misclassification cost, implying that pursuing a higher accuracy does not guarantee a lower cost. This shows that the value-based methods are indeed necessary, especially when the misclassification costs demonstrate a high degree of asymmetry under different error scenarios.

2.4. Experiments for Performance Evaluation

We conduct a series of experiments to evaluate the performances of the proposed methods against existing methods. The first set of experiments is based on a dataset for a real-world credit approval application downloaded from the UC Irvine machine learning repository (http://www.ics.uci.edu/~mlearn/MLRepository.html). Similar to the procedure adopted by Jiang et al. (2005), we preprocess the dataset to obtain a fully enumerated decision table that includes 15 binary attributes, and treat this table as the True Table for the entire population. From this True Table, we build the corresponding True Tree (TT). Given the True Tree, we can evaluate the performances of different methods.

2.4.1. Experiment Procedure

We first describe the experiment procedure for a fixed True Tree and a fixed set of parameter values, followed by the complete procedure that varies some of the parameter values. Additional experiments for robustness testing will be described in Subsection 2.4.4.

2.4.1.1. The Basic Procedure.

The basic experiment procedure for a True Tree is as follows:

Step 1. Obtain the underlying parameter values:

We first generate the *true* parameters for the underlying user population, including the percentage of liars, the liars' distortion matrices, the marginal distributions of the attributes for liars and truth-tellers. We then generate a random sample of users based on these parameter values. The sample of users is subsequently used to obtain the *estimated* distributions and distortion matrices.

Step 2. Generate KM Tree:

The KM Tree is constructed for comparison with the methods proposed in the present study. Since the KM method does not differentiate liars from truth-tellers, we construct the KM Tree using the marginal distributions and distortion matrixes for the entire population, similar to those illustrated in Figure 2.3(c) and 2.4(c).

Step 3. Generate ST, CT, VST, and VCT Trees:

Select an attribute as the verifiable attribute (VA). Following the procedures described in Sections 2.2 and 2.3, we construct the ST, CT, VST, and VCT trees.

Step 4. Simulate User's True and Observed Input Vectors:

Based on the marginal distribution of each attribute, we randomly generate a *true input vector* for each simulated user. This input vector is then fed into the True Tree to obtain

the *true recommendation*, which is used to decide the accuracy of other recommendations. Then, based on the percentage of liars in the population, we randomly determine whether the user is a liar or not. If the user is determined to be a truth-teller, her observed input vector is the same as her input vector. In case the user is a liar, her observed input values are distorted based on the assumed distortion matrices for liars.

Step 5. Calculate Accuracy and Misclassification Cost:

Feed each user's observed input vector into the True Tree, KM Tree, ST Tree, CT Tree, VST Tree, and VCT Tree to obtain their respective recommendations. Compare the recommendation obtained by each method with the true recommendation. If they are the same, the recommendation is considered correct, otherwise it is incorrect. For each incorrect recommendation, the corresponding misclassification cost is calculated based on the misclassification matrix illustrated in Figure 2.8.

Step 6. Compare Accuracy and Misclassification Cost:

Repeat steps (4) and (5) 10,000 times to simulate 10,000 users. Record the total number of correct recommendations and the total misclassification cost associated with each method. Compare the accuracies, i.e., the percentages of correct recommendations, and misclassification costs of different methods.

2.4.1.2. The Complete Procedure.

The basic experiment procedure uses a fix set of population parameter values, including the liar percentage and distortion matrices. To evaluate the performance of the

different methods under varying severity of lying, we control two important parameters in an extended set of experiments, i.e., liar percentage and distortion level for liars. The distortion level for an attribute measures the probability that the attribute's value will be distorted by liars, and is calculated based on the marginal distribution and distortion matrix for liars. The distortion levels are kept the same for all attributes in our experiments.

In these experiments, we try 11 different liar percentages varying from 0% to 100% and 9 different levels of distortion from 0.1 to 0.9. In addition, we randomly choose one attribute as the verifiable attribute (VA) during the expanded procedure. For a given combination of liar percentage and distortion level, we perform the basic procedure shown in subsection 2.4.1.1 to compare the performances of the different methods.

2.4.2. Experimental Results

Based on the data collected from the experiments, we calculate the average accuracy and the total misclassification cost of each method. The results are summarized in Table 2.3. As shown in the table, the CT method achieves an average accuracy of 73.2%, the highest among all the methods. The ST method places second, followed by TT (True Tree), VCT, KM, and VST. Regarding the misclassification cost, the VCT has the least overall misclassification cost at $56,953 for 10,000 simulated users. The VST ranks the second, and then CT, ST, TT, and KM.

Table 2.3 Performance Comparison Among Methods

Method	TT	KM	ST	CT	VST	VCT
Accuracy (%)	69.5%	69.1%	72.3%	73.2%	68.4%	69.3%
Misclassification Cost	87,122	90,011	79,846	76,882	59,277	56.953

To better understand how the performances of the different methods are affected by the severity of the users' lying behaviors, we plot their performances against two controlled parameters in the experiments, i.e., the liar percentage and the distortion level.

2.4.2.1. Liar Percentage

Figures 2.11(a) and 2.11(b) show the impact of liar percentage on accuracy under two distortion levels (50% and 70%, respectively). We find that CT always performs better than ST, KM, TT, VCT, and VST. When the percentage of liar is zero or 10%, CT, ST, VCT, and VST have the same accuracy. Their performance differences start to widen as the liar percentage increases. As the liar percentage approaches 100%, however, the performance differences between CT, ST, and KM again narrows. When the liar percentage is 100%, the three methods have exactly the same accuracy, since both ST and CT rely completely on the Liar Tree, which is the same as KM Tree under this condition. Similarly, VCT and VST have the same accuracy when liar percentage is 100% because they both provide the same recommendations as *Value-based Liar Tree*.

A surprising result observed from the two figures is that the True Tree can outperform the KM Tree when the liar percentage is not high. The advantage of True Tree over KM is more significant when distortion level is between 30% and 80% (Figure 2.11 (b)). This is a

result not observed in the study that develops the KM method (Jiang et al. 2005). We will further examine this interesting phenomenon in Subsection 2.4.3.

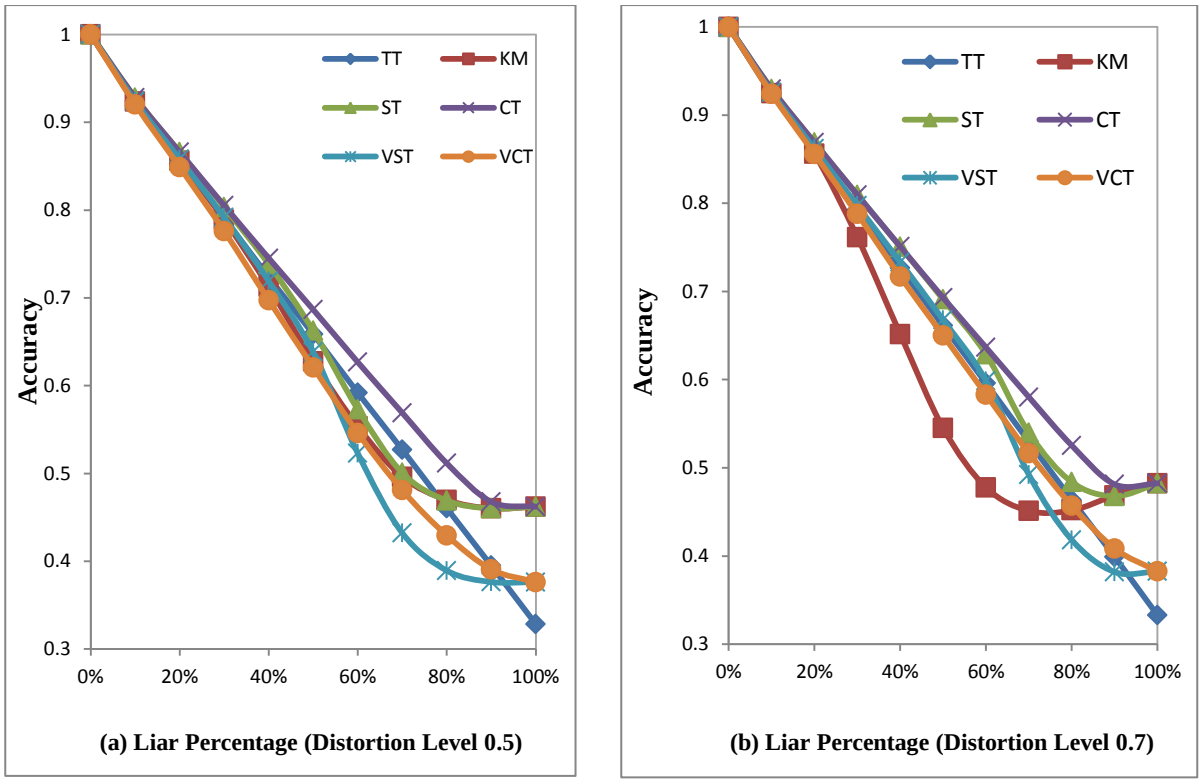

Figure 2.11 Impact of Liar Percentage on Accuracy

Figure 2.12(a) and 2.12(b) show the impact of liar percentage on the total misclassification costs at two distortion levels (0.5 and 0.7 respectively). We can see that the value-based methods lead to significantly lower misclassification costs than the other methods. Between the two value-based methods, VCT always performs better than or as well as VST in lowering the misclassification costs. The difference is smaller when the liar percentage is close to 100% or 0%. Regarding the other methods, we find that the cost-based performance ordering closely follows the accuracy ordering. For instance, CT has a relatively lower misclassification cost compared to ST, KM, and TT, and KM can lead to a higher

misclassification cost than TT unless the liar percentage is above 80%. In both cases, the lower (higher) misclassification cost is the result of a higher (lower) accuracy of recommendations.

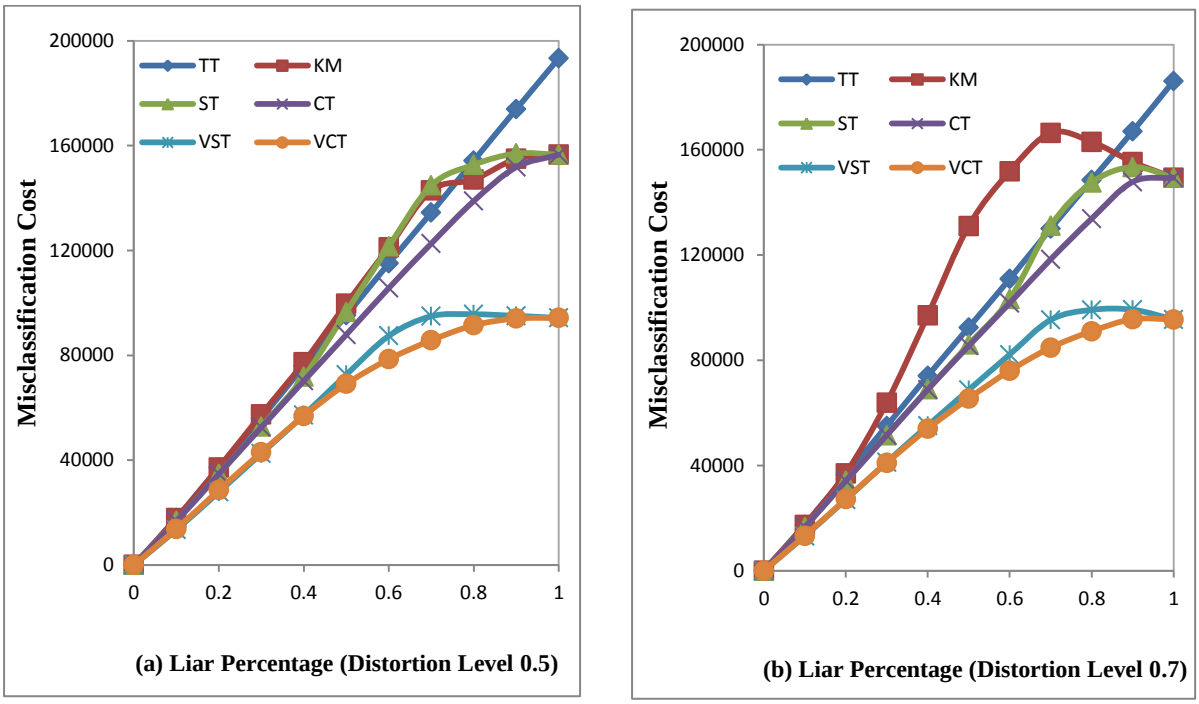

Figure 2.12 Impact of Liar Percentage on Misclassification Cost

2.4.2.2. Distortion Level

We next examine the impact of input distortion on the performances of the different methods. Figure 2.13(a) and 2.13(b) show its impact on accuracy under two liar percentages (50% and 70% respectively). We can see that CT still performs the best among all compared methods. When distortion level is very low (e.g., 0.1) or very high (e.g., 0.9), the performance difference between CT and ST narrows. Figures 13(a) and 13(b) also show that the True Tree (TT) has a clear advantage over KM unless the distortion level is low. At a higher distortion level, the KM can lead a worse accuracy than all other methods. Since the

two figures show the performances at a liar percentage of 50% and 70%, respectively, the results are consistent with the previous observation that the KM method does not perform well when the liar percentage is 80% or lower.

Further, by comparing Figure 2.13 with Figure 2.11 , we observe that the accuracies of the four methods proposed in this research, i.e., CT, ST, VCT and VST, generally decrease with the liar percentage, while they first decrease and then increase as the distortion level changes from very low (e.g., 0.1) to very high (e.g., 0.9). This is because when the distortion level is close to 1.0, there could be less than uncertainly about the true values. For instance, given a binary attribute with a distortion level of 0.9, then just choosing the value other than the observed value can lead to a 90% accuracy. The four proposed methods are sufficiently intelligent to incorporate this factor, hence their accuracies can improve as the distortion level is getting close to 1.0.

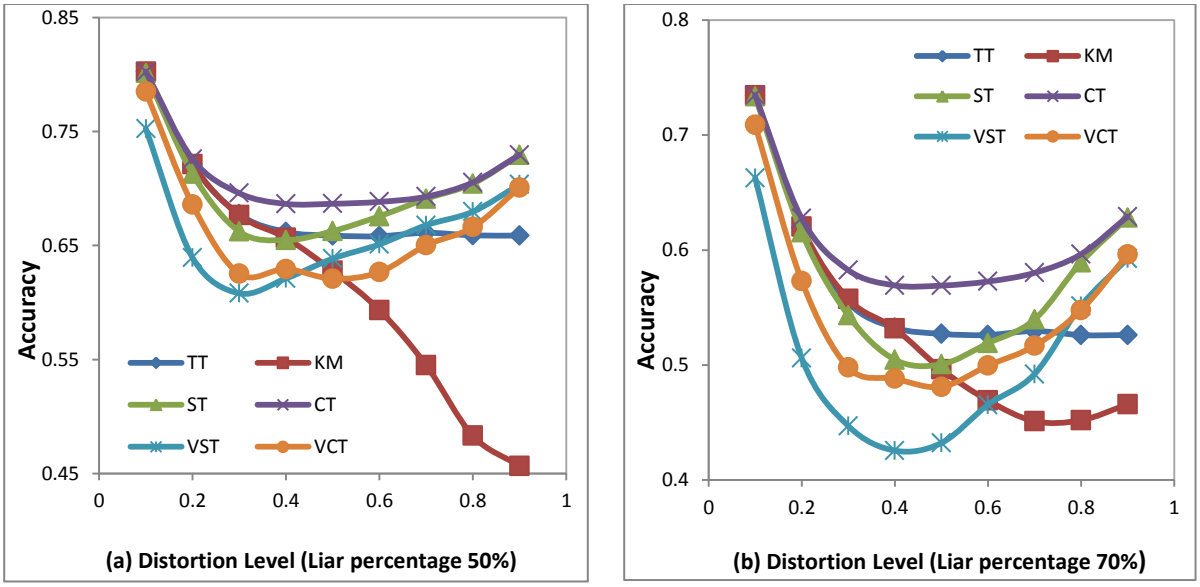

Figure 2.13 Impact of Distortion on Accuracy

The impact of the distortion level on the misclassification costs of the compared methods is shown in Figure 2.14 (a) and 2.14(b), corresponding to two fixed liar percentages at 50% and 70%, respectively. Again, VCT always has the lowest misclassification cost and VST places second. The performances of the accuracy-based methods again correlate with the accuracies of their recommendations, with CT performing better consistently better than ST. The KM method, again performs the worst when the distortion level is high.

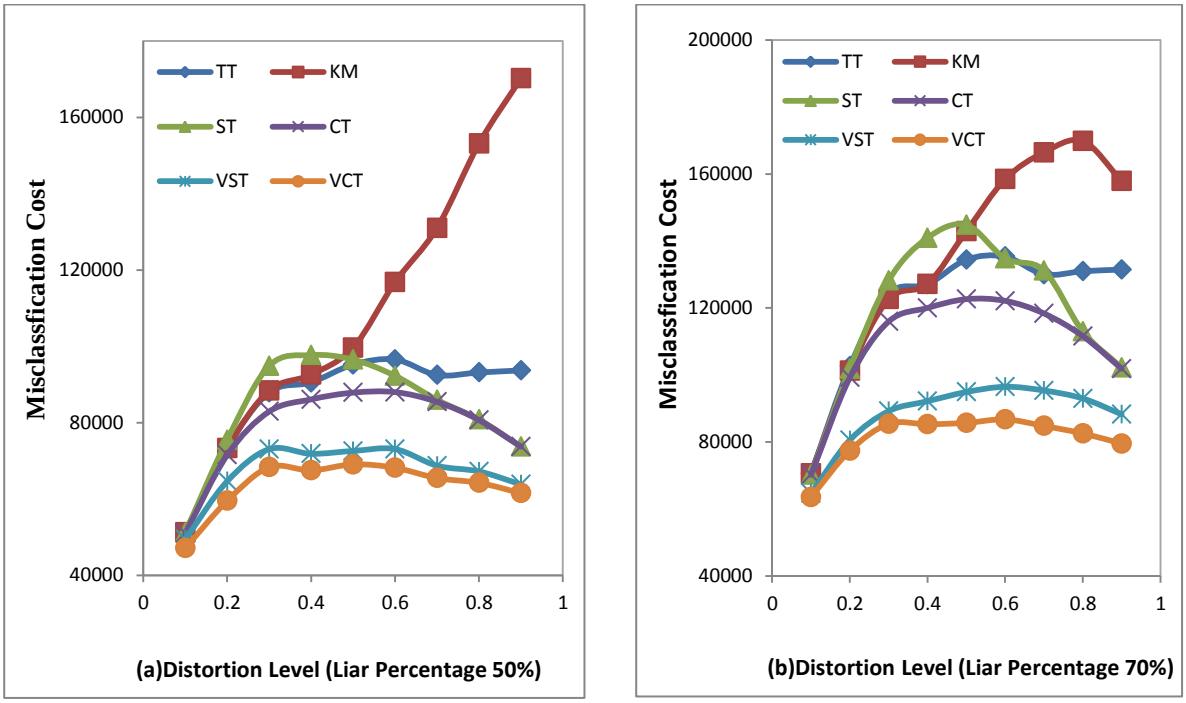

Figure 2.14 Impact of Distortion on Misclassification Cost

2.4.3. Why TT May Outperform KM?

As indicated in Subsection 2.4.2, KM does not show a consistent advantage over the True Tree (TT), a result that contrasts the finding by Jiang et al. (2005), which shows that KM consistently outperforms the True Tree. To understand why KM sometimes may perform worse than TT, we examine the composition of the user population that includes

both liars and truth-tellers. Note that KM does not differentiate liars from truth-tellers. Instead, it estimates the distortion matrices and marginal distributions based on the data for the entire population. In essence, the KM method assumes that all users may lie and the probability of lying about each attribute is the same across users.

In reality, there are users who tend to lie and users who rarely or never lie. We can illustrate this composition by Figure 2.15. There are two groups of users in a population. The first group accounts for 20% of the population and they lie 100% of the time. The second group accounts for 80% of the population and all of them are truth-tellers. When the two groups are mixed together, the liar percentage for the entire population is 20%. Now, if the True Tree is adopted for this particular population, although the recommendations for the liars may have a low accuracy, the recommendations for all truth-tellers are guaranteed to be correct. Because the truth-tellers account for 80% of the population, the average accuracy for True Tree will be at least 80%. On the other hand, when the KM Tree is adopted, since it assumes that everyone is a liar, the recommendations for truth-tellers may not be correct. Even the recommendations for liars may not have a high accuracy because the distortion matrices used by the KM, which is calculated for the entire population, are very different from those for the liars. As a result, the accuracy of KM may suffer. Therefore, the True Tree can outperform the KM Tree under this scenario.

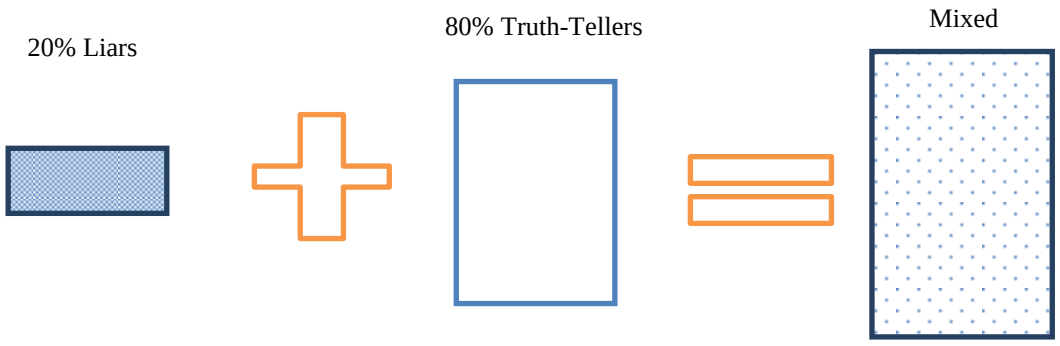

Figure 2.15 Mixing Liars and Truth-Tellers

Based on the user composition illustrated in Figure 2.15, we can imagine that when the percentage of liars in the population is high, the distortion matrices for the entire population, which are used by KM, will become similar to those for liars. Then, the performance of KM should improve. This is confirmed by Figure 2.11, which shows that the liar percentage is above 80%, the KM Tree clearly outperforms the True Tree, and it even matches the CT method when the liar percentage is close to 100%.

Based on experiment results and the analysis following Figure 2.15, we conclude that the KM method is recommended only if the user population is relatively homogeneous and the majority of them tend to lie. When there is a clear separation of truth-tellers and liars in the population and liars only account for a relatively small percentage of the population, the methods proposed in this study, CT and VCT in particular, should be used.

2.4.4. Robustness Tests

We conduct additional simulated experiments to evaluate the robustness of the proposed methods and examine whether the findings presented in Subsection 2.4.2 remain valid under different conditions.

2.4.4.1. Test on Different VAs.

In order to assess whether the selection of verifiable attributes (VAs) affects the performances of the proposed methods, we run some additional tests. With a fixed True Tree, we change the VA in each run, and record the performances of the proposed methods under different conditions. We find that there is no dramatic change in performances as the selected VA changes, and the findings presented in Subsection 2.4.2 remain valid qualitatively.

2.4.4.2. Test on Multiple VAs.

An intuitive extension of the proposed methods is to utilize multiple VAs to estimate the probability of a user being a liar. In additional experiments, we try two and three VAs on the ST and CT methods. For expositional convenience, we label the Double-VA extensions of ST and CT by DST and DST, and their Triple-VA extensions by TST and TCT, respectively. Figure 2.16 shows the how the performances of the different methods change with the liar percentage and distortion level. The average accuracies for ST, CT, DST, DCT, TST, and TCT are found to be 72.53%, 73.54%, 73.79% , 74.34%, 74.39% and 74.74%, respectively. These figures show that including more VAs slightly increases the accuracy, and the CT-based methods consistently outperform the ST-based methods. Our t-tests

confirm that the difference between ST-based and CT-based methods is significant regardless of the number of VAs used.

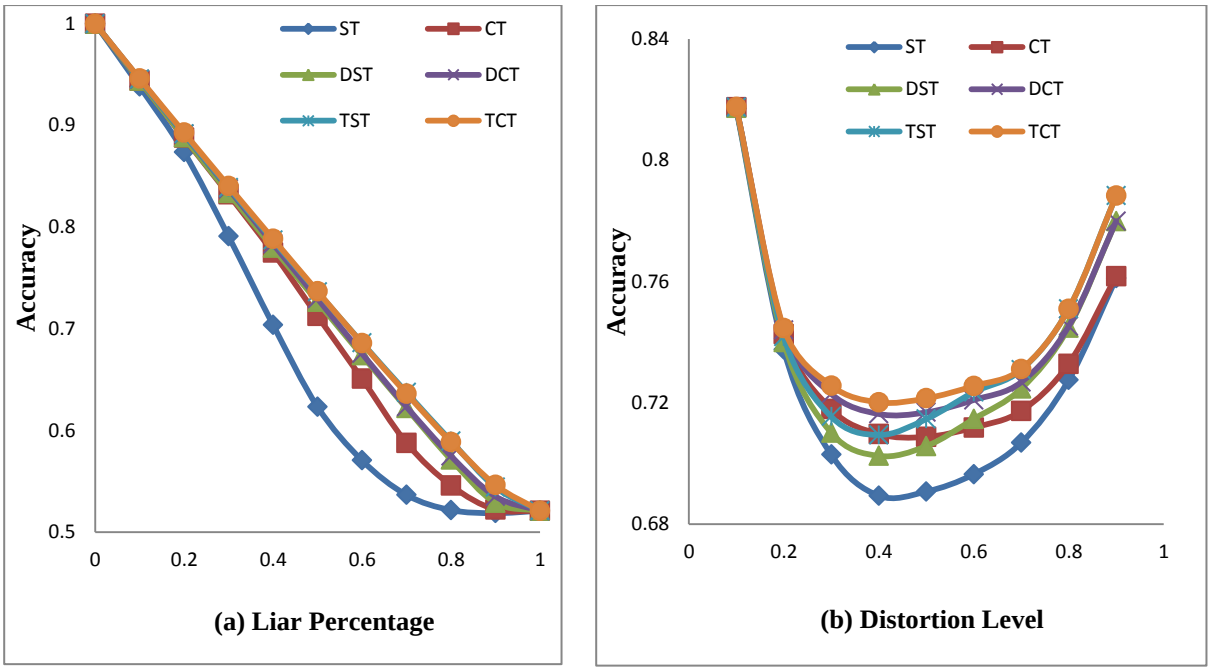

Figure 2.16 Multiple Verified Attributes Affect Accuracy

2.4.4.3. Test on Distortion Matrices.

We are also interested in evaluating whether the characteristics of the distortion matrices affect the performances of the compared methods. For this purpose, we adopt the Kullback–Leibler divergence (Kullback and Leibler 1951) to measure the degree of difference between two distortion matrices. The formula for this measure is

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_i P(i) \ln \frac{P(i)}{Q(i)} \qquad (6)$$

where P, Q are two matrixes and $D_{\mathrm{KL}}(P||Q)$ is the K-L divergence of Q from P. We select identity matrix as the benchmark matrix P. For each distortion matrix, we calculate its K-L divergence from the identity matrix. Then we explore the relationships between the K-L

divergence of each distortion matrix and the corresponding performances including both accuracy and misclassification cost. The results do not suggest a significant correlation between the Kullback–Leibler divergence and the performances of different methods.

2.4.4.4. Additional Tests based on Simulated Trees.

To further assess the performances of the different methods under different environments, in addition to the experiments based on the decision tree generated from the real-world credit approval dataset, we repeat a large number of experiments based on simulated decision trees. Specifically, similar to the decision tree for real-world credit approval application, we simulate 17 other true decision tables with 15 binary attributes. In addition, we simulated 100 true decision tables based on the credit risk assessment example shown in Table 1. In each of these tables, the recommendation corresponding to each input vector is randomly generated.

Similar to the experiments reported in Subsection 2.4.2, for each of the simulated True Trees, we again try 11 different liar percentages and 9 different distortion levels. After analyzing the experiment results, we conclude that the findings from these simulated experiments are largely consistent with the findings reported Subsections 2.4.2.

2.4.5. Summary

Based on the large number of experiments, we conclude that the four methods proposed in this study are superior to existing methods when users in the underlying population have varying degrees of tendency to lie. Among the four proposed methods, the Consolidated Tree methods (i.e., CT and VCT) are shown to outperform the Split Tree

methods (i.e., ST and VST) because CT and VCT are not only more accurate, but also lead to lower cost. Therefore, depending on whether the goal is to maximize accuracy or minimize the misclassification cost, either CT or VCT should be the method of choice for addressing the challenge of input distortion for deductive expert systems.

2.5. Selection of VAs

In this section, we discuss several practical issues related to the selection of verifiable attributes (VAs) to support the four methods proposed in this study (i.e., ST, CT, VST, VCT).

2.5.1. Determining the Best VA or VA Group

As shown in Section 2.4, the proposed methods outperform existing methods and can lead to significant benefits. Additional experiments reported in Subsection 2.4.4.2 demonstrate that performances of the methods can be further improved when multiple VAs are used. However, verifying the true values of a VA is not cost-free. Some attributes are less costly to verify than others. In the credit risk assessment example, for instance, verifying a user's bankruptcy status could be much less costly than verifying her income. When a group of VAs instead of a single VA is selected, although the method performances can be further improved, the verification costs will also increase.

Therefore, when deciding which attribute(s) to select as VA(s), we need to balance the benefit resulting from the reduction in misclassification costs (we can assume that the accuracy-based methods have a cost matrix with "0" on the diagonal and "1" elsewhere) and the costs of verifying the selected attributes. The attribute or group of attributes that leads to

the highest *net* benefit, which equals the expected *gross* benefit minus the cost of verification, should be selected as the VA(s) for the proposed methods. Since the gross benefit equals the reduction in misclassification cost, this is equivalent to selecting the attribute(s) that minimize the sum of misclassification cost and verification cost. We next discuss how the two costs can be computed.

The cost of verification for each selected VA or VA group equals the cost of one verification times the probability that the VA(s) will need to be verified at the time of consultation (recall that under CT or VCT, sometimes it is not necessary to verify the true values of VAs). The probability can be obtained by summing up the probabilities associated with all branches of a tree (e.g., a VCT Tree) that require the true values of the VA(s). In the credit risk assessment example, suppose the cost of verifying a user's *Bankruptcy* status is $20, and that of verifying *Education* is $45. When *Bankruptcy* is selected as the VA, the probability that it needs to be verified in the VCT true is 0.35. When *Education* is selected as the VA, the probability that it has to be verified is 0.41. Then, the expected costs of verification for the two attributes are

$$C_V(\text{Brankruptcy}) = 20 * 0.35 = 7.00, \text{ and}$$

$$C_V(\text{Education}) = 45 * 0.41 = 18.45.$$

The misclassification cost associated with each selected VA or VA group can be assessed based on either simulated or real performance evaluation. In either case, we need to obtain a *mis-recommendation matrix* for each selected VA(s), which shows how objects with a given true classification are misclassified into other classifications. In the credit risk

assessment example, for instance, the matrix should record the percentages of "low risk" customers that are incorrectly classified as "high risk" and "medium risk" by a proposed method. Once such mis-recommendation matrix is obtained, we can use it, along with the misclassification cost matrix and the marginal distribution of true classifications, to determine the expected *cost of mis-recommendations* corresponding to each selected VA(s).

	HR	MR	LR
HR	85%	14%	1%
MR	23%	71%	6%
LR	39%	18%	43%

Figure 2.17(a) Mis-Recommendation Matrix Corresponding to Bankruptcy

	HR	MR	LR
HR	87%	9%	4%
MR	22%	73%	5%
LR	31%	6%	63%

Figure 2.17(b) Mis-Recommendation Matrix Corresponding to Education

To illustrate, suppose the mis-recommendation matrices corresponding to the two VA options are shown in Figures 2.17(a) and 2.17(b), respectively. The misclassification cost matrix is the same as the one shown in Figure 2.8. The marginal probabilities associated with "HR," "MR," and "LR," are 0.1, 0.3, 0.6, respectively. The misrepresentation costs are calculated as

$$C_{MR}(\text{Brankruptcy}) = (45*14\% + 100*1\%)*0.1 + (10*23\% + 50*6\%)*0.3 + (20*39\% + 28*18\%)*0.6 = 10.024, \text{ and}$$

$$C_{MR}(\text{Education}) = (45*9\% + 100*4\%)*0.1 + (10*22\% + 50*5\%)*0.3 + (20*31\% + 28*6\%)*0.6 = 6.943.$$

Therefore, the total costs associated with the two VAs are

$$C_T(\text{Brankruptcy}) = C_V(\text{Brankruptcy}) + C_{MR}(\text{Brankruptcy}) = 17.024, \text{ and}$$

$$C_T(\text{Education}) = C_V(\text{Education}) + C_{MR}(\text{Education}) = 25.393.$$

Based on this result, *Bankruptcy* is a better option than *Education* as VA. Similarly, the costs associated with other feasible VA(s) can be computed. The VA(s) with the lowest total cost should be selected.

2.5.2. Selecting an External VA

In some cases, it may be very difficult or costly to verify any of the *internal attributes*, i.e., those included in the True Tree. As an alternative, we could use an *external attribute*, one that is not included in the True Tree but easier to verify, to estimate the probability that a user is liar. For instance, in the credit risk assessment example, a user's *Citizenship* status is not included in the True Tree. We next illustrate this attribute can be used as an external VA to help implement the VCT method.

Citizenship has two possible values (*Yes*, *No*). Similar to interval attributes, we can obtain the distortion matrices and marginal distributions for this external VA, as illustrated in Figures 18(a) and 18(b). Based on these distortion matrices and marginal distributions, once a user's observed and true *Citizenship* status is known, we can calculate the probability that a user is a liar or not. Analogous to the basic steps for the VCT method, we can built an extended VCT Table with the observed and true values of *Citizenship* included, and then construct an extended VCT Tree from this table. Similarly, the other three proposed methods can also be extended to include the external VA.

Citizenship (C) for Liars

	Yes	No
Yes	0.39	0.61
No	0.17	0.83

Citizenship (C) for Truth-Teller

	Yes	No
Yes	1.00	0.00
No	0.00	1.00

Figure 2.18(a) Distortion Matrices for the External Verified Attribute

Citizenship (C) for Liars

Yes	0.52
No	0.48

Citizenship (C) for Truth-Teller

Yes	0.6
No	0.4

Figure 2.18(b) Marginal Distribution for the External Verified Attribute

Using external VAs does not affect the proposed methods in the same manner. For Split Trees, the tree structure is not much affected since the VA is always first verified regardless of whether it is an internal one or not. For Consolidated Trees, since the observed and true values for an external VA are treated as two additional attributes in the extended decision table, the tree structure changes.

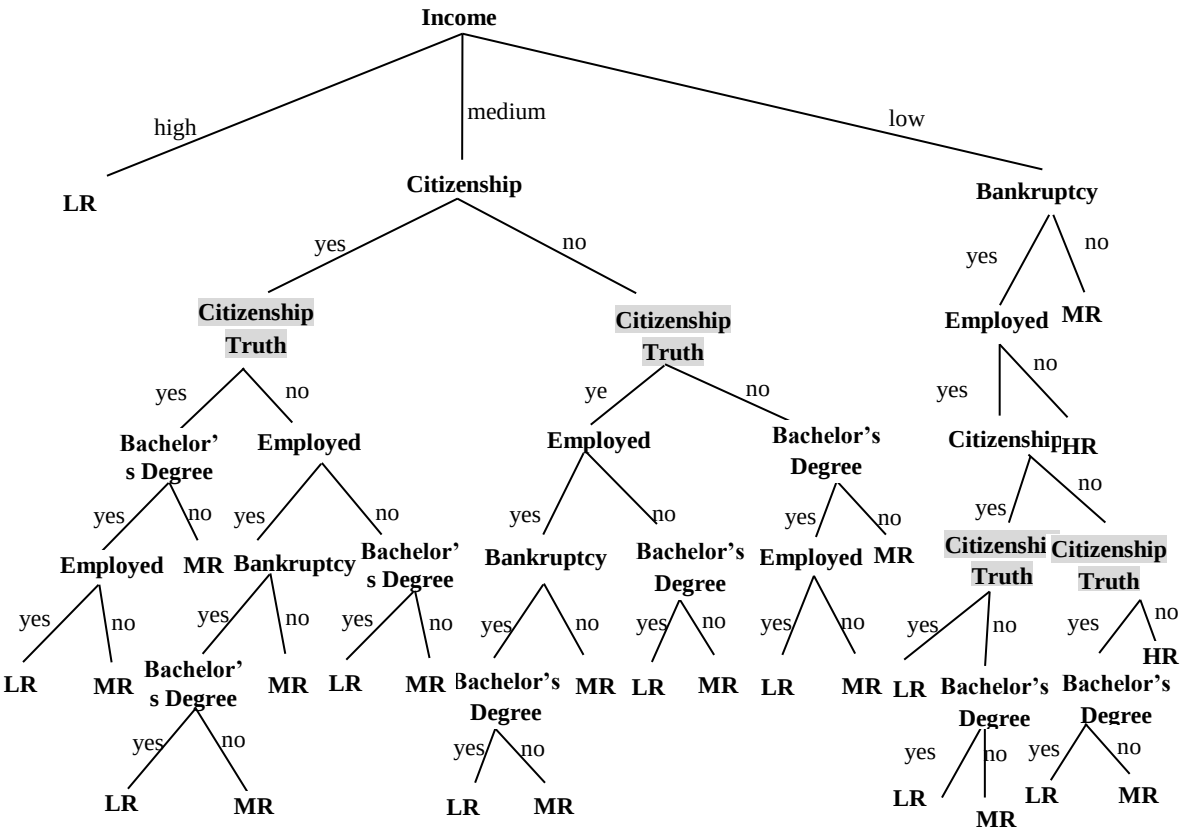

Figure 2.19 Consolidated Tree with External Verified Attribute

With *Citizenship* as an external VA, and using values shown in Figures 2.2, 2.3, 2.4, and 2.18, we obtain a Consolidated Tree for the credit risk assessment example, as shown in Figure 2.19. Compared to Figure 2.7, this tree changes dramatically with more nodes and layers. Note that in certain branches, we do not need to verify the value of the external VA, which is consistent with CT Trees built using internals VAs.

Still with *Citizenship* as an external VA, we build other trees and run performance evaluations. Figure 2.20 shows the how the performances of the different methods change

with the liar percentage and distortion level. The average accuracies for TT, KM, ST, CT, VST, and VCT are found to be 71.89%, 71.43%, 73.88% , 74.88%, 69.54% and 70.62%, respectively. The results and the figures show that the performance comparison across methods is largely consistent with the results obtained when internal VAs are used.

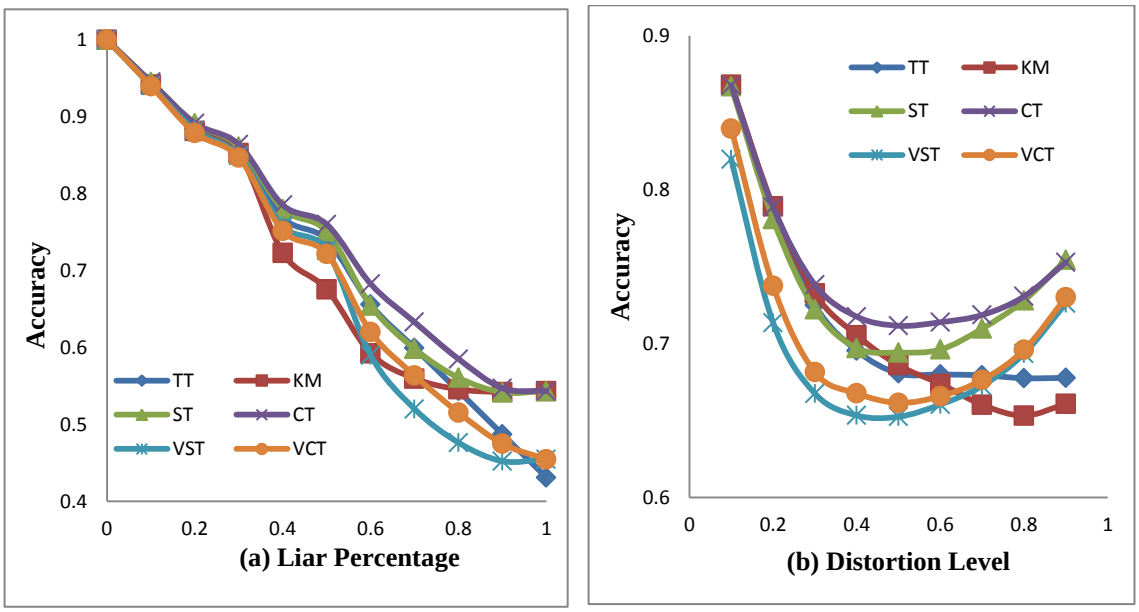

Figure 2.20 External Verified Attributes Affect Accuracy

The fact that an external VA can be as effective as internal VAs brings more options when selecting the VAs for the proposed methods. It is even possible that a combination of internal and external attributes can be adopted. The wider set of choices can lower the barrier for implementing the proposed methods and potentially reduce the verification cost by a substantial amount.

2.5.3. Dealing with Strategic Agents

Once the proposed method (e.g., VCT) deployed, it is possible that after repeat tries, a strategic agent or groups of such agents can find out which attribute is used as a VA. Subsequently, they could try to "defeat" the system by providing true values to the VA, but lying about other attributes. There are a number of countermeasures to deal with such strategic behavior. The first option is to change the VA from time to time. The second one is to randomize the selection of VAs at runtime. We have built and tested such variation methods and found that the performance comparisons of the different methods remain largely unchanged from the cases with fixed VAs.

2.6. Discussions and Future Research Directions

Despite the prevalence of input distortion by users of expert systems, research on how to effectively address such challenges has been limited. The methods proposed in this study explicitly differentiate liars from truth-tellers and treat them differently when their information is provided to redesign deductive expert systems. Two of the proposed methods, i.e., Split Tree (ST) and Consolidated Tree (CT), attempt to improve the accuracy of recommendations, and the other two, i.e., Value-based Split Tree (VST) and Value-based Consolidated Tree (VCT), aim to minimize the expected misclassification cost resulting from incorrect recommendations. Experimental results show that the proposed methods can lead to significantly better accuracy or lower cost than existing methods. Between the two pairs of proposed method, we find that CT consistently outperforms ST in improving the accuracy of recommendations, and VCT always performs better than VST in reducing the expected misclassification cost. In addition, we find that the KM method proposed by Jiang et al.

(2005), which essentially assumes that all users are potentially liars and treats them in the same manner, is not effective when there is a clear separation of liars and truth-tellers in the underlying population, a finding that further confirms the necessity of differentiating liars from truth-tellers when addressing input distortion by users.

The methodologies and findings of this study have important financial implications. Although the proposed methods require the verification of user-provided attribute values, the cost of such validation can be controlled by selecting the attributes that are relatively easy to verify. As a result, the expected benefit of adopting the proposed methods should exceed the expected cost under most real-world applications. Given the wide application of expert systems in various problem domains, the proposed methods can potentially lead to significant financial saving for organizations. Furthermore, providing both accuracy-based and value-based methods gives firms the flexibility to select the appropriate method based on the underlying misclassification cost structure. Specifically, when the misclassification cost matrix is asymmetric, the VCT method is most preferred. When the misclassification cost matrix is approximately symmetric or the misclassification costs are very difficult to estimate, the simpler CT method should be adopted.

The methods we propose in this study can be extended in future research to address more complex problem scenarios. For instance, in this study we consider only two groups of potential users, i.e., liars and truth-tellers. Such two-group model may not be sufficient to capture the heterogeneity among potential users. For instance, some users never lie, some may lie occasionally, while others who lie frequently. How to extend our model to

incorporate multiple groups of users is an interesting direction for further research. Furthermore, the proposed methods are computationally intensive, hence they may become impractical when the number of attributes or the number of states for the attributes are large. In a future study, one could simplify the methods to reduce their complexity. For instance, when computing the CT Table, it is possible that the accuracy may not degrade much even if we consider only a small subset of the possible true vectors given an observed vector. The computation time can be significantly reduced if a subset of vectors can be identified and used in constructing the trees.

CHAPTER 3. DEVELOP PROFITABLE RECOMMENDATION SYSTEMS WITH SHILLING ATTACK DETECTION

3.1. Introduction

With the significant advances in web technology, there has been explosive growth of electronic transactions. A flood of product information including consumer reviews or ratings are now available on the Internet. Consumers nowadays have various and easy access to different products through online searches. This type of information overloading has presented great challenges to the customers. Asch (1961) found that when customers selected the items for the first time, they would tend to rely on other customers' opinions. Accordingly, recommendation systems have been designed to help customers to identify their favorable items. The rationale behind an important technique (e.g. collaborative filtering) commonly used in recommendation systems is that customers who share similar tastes in the past tend to have similar tastes in the future. For a particular customer (the active user), collaborative filtering recommendation systems will select a list of users who share the similar preference with the active user. The selected users are called neighbors and the predicted preference of the active user is based on the ratings of their neighbors. Hence, many electronic retailers (e-retailers) such as Amazon and Netflix have attempted to harvest this big data through recommendation systems by predicting users' preferences through business analytics based on users' neighborhood selection, opinions collections, etc. (Herlocker et al. 1999; Huang et al. 2007; Lam and Riedl 2004).

In order to improve the quality of recommendations, it is very important to identify the proper neighbors. Traditional collaborative filtering recommendation systems have

attempted to identify users' neighbors by focusing on their preference similarity with the active user. Despite the popularity of this approach, it suffers from at least two limitations. First, since the preference is measured by the historical ratings, traditional recommendation systems are vulnerable to attackers who deliberately seek improper benefits by injecting fictitious ratings. For example, one e-retailer (e.g. Amazon) may have various products belonging to third party corporations on its electronic marketplace. Attackers may promote their own products to make them more frequently recommended (Lam and Riedl 2004). These attacks as termed as "shilling attacks" and this behavior may affect other e-retailers' profit negatively (Burke et al. 2005, Lam and Riedl 2004, O'Mahony et al. 2004). Previous studies in shilling attack detections have discovered unusual ratings patterns of attacks by examining an individual or a group of attackers' profile (Chirita et al. 2005; Lee and Zhu 2012). Despite the overall advantage of attacker detection in the previous studies, the precision is very high when attack size or filler size is relatively large (Lee and Zhu 2012; Williams et al. 2007). In contrast, when the attack size or filler size is small, their performance deteriorates. The false positive rate is high since the attackers are not easily distinguishable from genuine users. Hereby, it indicates a high possibility of misclassifying genuine users as attackers. Since previous detection methods usually use the filtering-based strategies, which means that the classified attackers will be removed from the database, misclassifying a genuine user may influence the recommendation quality permanently.

The second limitation is that none of the recommendation systems with shilling attack detections takes the profit of e-retailers into consideration. Predicting customers' preference accurately may not be enough for e-retailers since various products have different profit

margins. From the e-retailer's perspective, the ultimate goal of applying recommendation systems is to increase profit (Das et al. 2010). The importance of the e-retailer's profit has been noticed by previous scholars. However, most of the approaches still rely on neighbors' preference by incorporating product profitability as a weight to generate recommendation. Each user is assumed implicitly as a genuine user. Once the neighbor's preference is distorted by attackers, the recommendation quality is also affected negatively.

In this study, we propose an improved approach to address the above limitations. We intend to integrate the profitability factor into the traditional systems under the attack environment. For each active user, we attempt to provide quality recommendations by selecting the proper users' neighbors that control the influence from attackers while maximizing the e-retailer's expected profits. We start by examining each user's potential as an attacker. To avoid the loss from misclassifying genuine users, especially when either filler size or attack size is small, we adopt the discounting-based strategy to deal with the suspicious user. An attack probability is estimated for each user and his or her rating will be discounted based on this probability. We then find the optimal neighbor set for each active user, which maximizes the e-retailer's expected profits constrained by the controlled shilling attacker level within a certain threshold. Generally, we intend to select the neighbors whose recommendations can increase the e-retailer's expected profits and those who have no or little chance of being classified attackers. Finally, the selected neighbors are used to generate predicted ratings for active users on each unrated item.

We conduct an experimental study based on a real world database to show the effectiveness of our proposed approach. We compare our approach with two benchmark methods and experimental results indicate that our proposed approach can increase the e-retailer's expected profits without losing the recommendation accuracy. The proposed approach has made significant contributions to designing and implementing a profitable trustworthy recommendation system. To the best of our knowledge, this research is the first effort that aims to provide a value-based neighbor selection (VNS) that strikes a balance between the consumers and retailers. From customers' perspective, the recommendations from such neighbors are more helpful since they resist the influence from the attackers. From e-retailers' perspective, such recommendations are more accurate to attain customers and they are more profitable.

The rest of the paper is organized as follows. In section 3.2, we provide a brief review of research on recommendation algorithm, followed by an overview of the literature on shilling attacks and profitability-based recommendation systems. In section 3.3, we propose our value-based neighbor selection approach with detailed analysis. Section 3.4 summarizes experimental findings. Finally, section 3.5 concludes this study with implication discussion and potential directions for future work.

3.2. Literature Review

3.2.1. Collaborative Filtering Recommendation Algorithm

Traditional collaborative filtering recommendation algorithm is based on Pearson Correlation Coefficient (PCC) (Breese et al. 1998), which is one of the most commonly used

algorithms. Data is represented as an $(n \times m)$ user-item ratings matrix. The similarity $w(a,i)$ between user a and i is calculated as:

$$w(a,i) = \frac{\sum_{j \in J}(r_{a,j} - \bar{r}_a)(r_{i,j} - \bar{r}_i)}{\sqrt{\sum_{j \in J}(r_{a,j} - \bar{r}_a)^2 \sum_j (r_{i,j} - \bar{r}_i)^2}} \quad (1)$$

where r_{ij} is a rating score of item j expressed by user i, $\bar{r}_i$ is user i's average rating calculated from the set of items on which user i has recorded ratings, and J is the set of items the user a and i both rated.

The predicted rating $p_{a,k}$ for user a on a specific item k is computed as:

$$p_{a,k} = \bar{r}_a + \kappa_a \sum_{i=1}^{n} w(a,i)(r_{i,k} - \bar{r}_i)\, sd \quad (2)$$

and $\kappa_a = 1/\sum_{i=1}^{n} |w(a,i)|$ is a normalizing factor, forcing the predicted rating to be within the given range of $r_{i,k}$. Hence, the prediction given by the user-based collaborative filtering algorithm is correlated with the weighted average of other users' preferences.

3.2.2. Shilling Attack Models

There are two types of attacks. One is the push attack, which promotes the target items by maximizing their ratings; the other type is the nuke attack, which denotes the target items by minimizing their ratings (O'Mahony et al. 2004). Hereby, the predicted ratings on the target items are expected to be higher in the push attack and lower in the nuke attack.

Figure 3.1 shows the general attack item-rating profile, which is adapted from Mobasher et al. (2007). The first row lists each item *i* in the entire set *I* and the second row

displays the corresponding ratings. Items could be categorized into four exclusive sets: the *selected* items set$I_S$with rating denoted as $\delta(\cdot)$, the filler items setI_F with rating denoted as $\sigma(\cdot)$, the unrated items set $I_N$with null ratings and the target items set $I_T$with rating denoted as $r(i_t)$. Four types of attack model on both push and nuke attack by injecting malicious profiles have been found. We discuss the general characteristics of each model below. Detailed information about the attack models can be found in Mobasher et al. (2007).

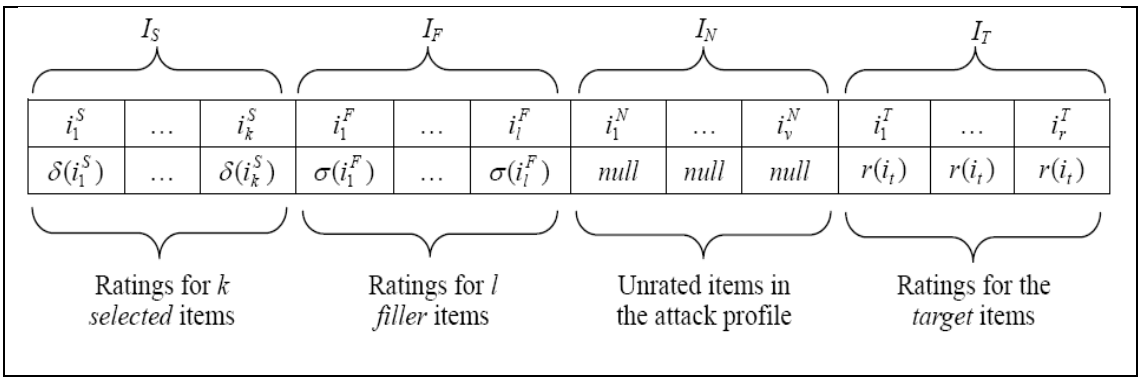

Figure 3.1 The Structure an Attack Profile

- *Random Attack*: For attack profile in random attack model, its selected items set $I_S = \emptyset$. For filler items, $\sigma(i_j^F), j = 1, \cdots, l$ are generated across all attack profiles from one identical normal distribution that consists of the global mean and the global standard deviation across all users and all items in *I*. This type of filling method is named as the *Random Filling Method (RFM)*. $r(i_t)$ is assigned to r_{max}, the maximum allowed rating in the system, if it is push attack. For nuke attack, $r(i_t)$ is assigned with r_{min}, the minimum allowed rating in the system.

• *Average Attack*: $I_S = \emptyset$ in average attack model. For filler items, $\sigma(i_j^F), j = 1, \cdots, l$ are derived from an individual normal distribution consists of the mean and standard deviation for item *j* across all users. This type of filling method is named as the *Average Filling Method (AFM)*. For target items, $r(i_t)$ is assigned with r_{max}/r_{min} if the attack type is push/nuke.

• *Bandwagon Attack*: The bandwagon attack aims to increase the similarity between attack profiles and a large number of users by incorporating the most popular items across all users intoI_S. Thus, $\delta(i_1^S) = \cdots = \delta(i_k^S) = r_{max}$. $\sigma(i_j^F), j = 1, \cdots, l$ are generated as in the random attack.

• *Segment Attack*: The segment attack has the same structure as the bandwagon attack. But it is designed to attack a group of users with the known preference. The selected items in I_S are from a target set of users with similar tastes. For instance, within a segment of book preference (e.g. fairy tale), a bunch of popular books could be selected and users who favor this type of book are more likely to be affected. Accordingly, if the size of I_S is larger, though increasing the risk of detection, the attack profiles will be more effective since each item can affect its own segment of users. If the items in I_F are assigned with r_{min}, the segment attackers can have best impact, while their risk of detection does not decrease either.

3.2.3. Attack Detection

Attack detection has been studied prolifically by researchers. One stream is to develop more robust and secure recommendation systems to withstand attackers. Hofmann

(1999) has developed the Probabilistic Latent Semantic Analysis (PLSA) method to characterize hidden semantic associations among objects. This method is applied by Jin et al. (2004) to create user cluster based on web data and also by Mobasher et al. (2003) to the collaborative filtering. In both studies, applying PLSA increases the accuracy and stability over the traditional collaborative filtering algorithms.

The second stream of research deals with anomaly detection in time series. Keogh et al. (2006) and Zhang et al. (2006) have explored the problem of locating discords. Brutlag (2000) has integrated an exponential smoothing model and Holt-Winters forecasting into the Cricket architecture for network traffic analysis. Lakhina et al. (2005) have found using packet features distributions could also facilitate detection on network traffic analysis.

Another stream of research aims at filtering attack profiles. Mehta et al. (2007) detect a set of attackers using principal component analysis with the assumption that the attacker number is known. Williams et al. (2006) discuss how to dealing with attackers when their profiles are obscured. Chirita et al. (2005) propose metrics that can measure and identify the attacker's profile before developing recommendations. Williams et al. (2007) adapted certain metrics and proposed the detection method names as Classification-Based Approach (CBA). Since the CBA need to train classifier, attacks with unknown types are not applicable. Lee and Zhu (2012) introduced a two-phase procedure for attack detection by analyzing the difference between attacks and genuine users as a group. Despite that the detection accuracy improves significantly, the precision is still not satisfied.

3.2.4. Value Based Collaborative Filtering

The profitability of recommendation systems for e-retailers has been noticed by scholars in recent years. Shani et al. (2002) proposed a Markov decision processes (MDP) model that took the expected profit of the recommendation into consideration. Despite the commercial value of this approach, it is not applicable when we do not know the chronological order for customers to select the recommended item. Liu and Shih (2005) proposed that recommendations could be generated based on the customer lifetime value, which is evaluated by customers' recency, frequency, and monetary (RFM). This method requires a detailed record of customers' historical transactional time and behavior. When relationship information among users could be obtained, Akoglu and Faloutsos (2010) adopted a social network analysis to provide more accurate and profitable recommendation. When the recommendation system has an interaction opportunity with the customer, Jiang et al. (2011) developed a dynamic pricing strategy to recommend products while maximizing e-retailer's profit. Das et al. (2010) proposed a theoretical model that maximized the e-retailer's profit while maintaining the similarity between the recommendation and customers' real preference. However, they assumed that an accurate recommendation should already be predicted by other existed recommendation systems. Chen et al. (2007) designed a model called *Hybrid Perspective Recommender System (HPRS)* that always recommended the most profitable item to the customer. Though this approach is applicable in most cases, they do not consider the vulnerability under attacks.

3.3. Value-Based Neighbor Selection

In this study, we propose a novel value-based neighbor selection (VNS) approach based on users' similarity, the expected profitability of users' recommendation, and users' attack probability. This method is designed for user-based collaborative filtering recommendation systems and it attempts to maximize e-retailer's profit while coping with the attackers. Only push attack is discussed in this study. In this section, we describe the proposed approach in details.

3.3.1. Problem Definition

Given an active user, VNS method can be applied to form a proper group of users who play a role in generating recommendations. We call the selected users as the neighbors of an active user. Hence, the inputs to the VNS method include two types of data. One is user information, e.g., historical user-item rating matrix. The other is item information, e.g., item profit. The outputs from the VNS method are lists of selected neighbors.

Denote a user as u_k with k=1 , . , . K. The rating for u_k on item i_j is denoted as $r_{k,j}$. Given the user's historical rating pattern, we can evaluate the attacker probability for u_k as $s_k \in (0, 1)$. The evaluation process will be discussed in details in the next subsection.

Let J be the total number of items in the database and i_j stands for the item j, where j=1 , . , . J. For each item i_j, its unit market price and its unit cost are denoted as m_j and o_j respectively, j=1 , . , . J. The corresponding unit profit for item i_j, denoted as p_j, is calculated as the difference between m_j and o_j, j=1 , . , . J.

For each active user u_t in need of recommendation on items $i_l \sim i_n$, where l<n<J, we intend to predict u_t's rating on those items, based on u_t's neighbors' rating on them as well as the similarity or the relative influence (weight) from another user u_k on user u_t. The similarity is denoted as $w(t,k)$. It can be calculated based on u_t and other users' historical rating through (1). Finally, the decision variable $\boldsymbol{x^t} = [x_1^t, x_2^t, \ldots x_k^t, \ldots x_K^t]$ denotes the neighbor selection list for active user u_t, where $x_k^t = 1$ if the user u_k is selected as a neighbor and $x_k^t = 0$ when the user u_k is not selected.

3.3.2. The Proposed Method

When selecting neighbors for each active user, the VNS method takes into account of both the active user's preference and the e-retailer's profit. More specifically, it follows two principles:

1) To predict u_t's preference more accurately, the influence from potential attackers should be limited.

2) To maintain e-retailer's profit, the expected profit of the recommendation generated from the selected neighbors for u_t should be maximized.

Hereby, our goal is to maximize the e-retailers' expected profit while controlling attackers level in the neighbors below a threshold τ. The shilling attacker level, which is defined as the mean of the number of attackers in the neighbor list, could be evaluated as the sum of each user's attacker probability. Since in the electronic systems, users are independent from each other and each user has its own probability to be an attacker. Thus

the attacker distribution in the entire user population follows a Poisson binomial distribution. Accordingly, we discuss the process of evaluating attacker's probability and the expected e-retailer's profit in the following subsections.

3.3.2.1. Shilling Attacker Probability

To estimate the probability of each user as an attacker, we should first discover the characteristics of attackers. Previous scholars have already conducted a series of studies on shilling attack detection based on attackers' rating pattern (Chirita et al. 2005; Lee and Zhu 2012; Mobasher et al.2007). Two features of attackers' rating have been widely recognized. First, the extraordinary ratings pattern of attack profiles has been attributed and metrics to measure such a pattern has been developed. For example, as discussed in Chirita et al. (2005), *Rating Deviation from Mean Agreement* (RDMA) is adopted to identify attacker by examining the user profile's deviation of agreement with other users on a set of items. However, this approach is not successful in the case of larger attack size (Chirita et al. 2005). Lee and Zhu (2012) proposed the metric as *Group Rating Deviation from Mean Agreement* (GRMDA), which measures each cluster of user profile's deviation from the overall mean agreement. The GRDMA is measured by

$$GRDMA_C = \max\left\{\left|\bar{r}_{C,j} - Avg_j\right| \cdot \left(\frac{NR_j^C}{NR_j}\right)^2\right\},\ j = 1, \cdots, N_C\ , \tag{3}$$

where N_C is the number of items rated by users classified into cluster C; $\bar{r}_{C,j}$ is the average of the ratings for item j rated by users classified into cluster C; NR_j^C is the amount of ratings of item j rated by users classified into cluster C; Avg_j is the average of the ratings for item j rated by all users, and NR_j is the amount of ratings of item j rated by all users;

The basic rationale behind applying GRMDA is that the target items of attackers are usually unpopular in push attack. Thus, the target items have only a few, but high rating so that the attackers are inclined to have unusual ratings. Hereby, once such users are found within a cluster, the GRMDA value will be higher than clusters of genuine users. Details can be found in Lee and Zhu (2012).

Second, attackers are more likely to behave similarly to other users in the system in non-target items so as to affect recommendations effectively (Chirita et al. 2005; Lee and Zhu 2012; Mobasher et al.2007). The more similar between two users, the smaller the distance between them is in the space. Thus, this feature can be evaluated by calculating the distance between one user and all other users by adopting Multidimensional Scaling (MDS) approach (Lee and Zhu 2012). The user with a smaller average distance has a higher possibility as an attacker.

Therefore, in this study, we adopt GRMDA along with average distance to estimate the probability of each user as an attacker. The basic procedure is depicted below.

Step 1: $GRDMA_C$ Calculation.

Following Lee and Zhu (2012), calculate $\mathrm{GRDMA_C}$ for each user cluster. Select the user cluster $\mathrm{C_{max}}$ which has the highest GRMDA. For users not in the cluster $\mathrm{C_{max}}$, their shilling attack probability is 0 ($s_k = 0$, k=1 , . , . K, & $\mathrm{k} \notin \mathrm{C_{max}}$).

Step 2: User-User Distance Calculation.

For all users in the cluster C_{max}, calculate user-user distance in g-dimensional MDS space based on their user-rating matrix. For each user, summarize and average its distance with all other users in the system, denoted as $dist_k$ (k=1 , . , . K & $k \in C_{max}$).

Step 3: Attacker Probability Estimation.

Find the mean and the minimum of all users' average distance ($dist_{mean}$ and $dist_{min}$) in the cluster C_{max}. Then normalize and translate the values in the average distance into an attack probability in value [0, 1]. We choose the transformation approach as shown in (4) such that the user with the lowest average distance has the highest probability to be an attacker. Also, the users with very high average distance (i.e. greater than the mean) are assigned 0 for attacker probability.

For $[dist_{min},\ dist_{mean}]$ -----> [0, 1]

$$s_k = 0 \quad \text{if } (dist_k > dist_{mean});$$

$$s_k = \frac{1}{(e-1)}\left(e^{\frac{dist_{mean}-dist_k}{dist_{mean}-dist_{min}}} - 1\right) \qquad \text{otherwise;} \tag{4}$$

3.3.2.2. Expected E-retailer's Profit

For the active user u_t , the expected e-retailer's profit obtained from recommendation systems is given by $E_t = \sum_{j=l}^{j=n} p_j * r_{t,j}$, where $r_{t,j}$ is the predicted rating for active user u_t on item i_j and p_j is the profit for item i_j. Moreover, in the user-based collaborative filtering recommendation systems, the active user's predicted rating is correlated with the neighbors'

rating and the similarity between u_t and neighbors. Accordingly, the expected e-retailer's profit for u_t from the selected neighbors is equivalent to

$$E_t = \sum_{j=l}^{j=n} p_j * \sum_{k=1}^{k=K} r_{k,j} * w(t,k) * x_k. \quad (5)$$

Therefore, by taking into account of the shilling attacker level as well as the expected e-retailer's profit, the VNS method for the user-based collaborative filtering recommendation systems is proposed as follows:

$$Maximize \ \sum_{j=l}^{j=n} \sum_{k=1}^{k=K} r_{k,j} * w(t,k) * x_k * p_j \quad (6.1)$$

$$s.t. \ \sum_{k=1}^{k=K} s_k * x_k < \tau \quad (6.2)$$

$$\sum_{k=1}^{k=K} x_k > 1 \quad (6.3)$$

This approach is designed to maximize the profits of the e-retailer when selecting neighbors to generate recommendation for each active user (6.1) while maintaining the level of shilling attackers existed in the selected neighbors is below the controlled level (6.2). In addition, constraint (6.3) makes sure that at least one neighbor is selected to each active user at each time. Since x_k is a binary variable that indicates whether or not a user is selected as a neighbor, we can identify the potential undesirable outcome. That is, if a user's individual attack probability is higher than τ, it is always excluded from the neighbor list.

3.3.3. An Illustrative Example

Now, we present step by step the operations of the method proposed above through an illustrative example. There are two types of inputs to the VNS method, 1) the item sample profit array, as shown in Table 3.2; 2) the users' item rating matrix, as shown in Table 3.1.

Table 3.1 User-Item Rating Matrix

$r_{k,j}$	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	i_{17}	i_{18}	i_{19}	i_{20}	i_{21}
u_1	8	0	0	8	0	7	7	8	0	0	0	5	10	0	0	8	8	0	0	0	0
u_2	0	1	0	0	7	0	0	0	0	8	0	0	3	3	0	0	2	1	0	0	0
u_3	0	0	6	0	4	0	7	4	0	0	0	3	0	0	3	4	0	0	3	0	2
u_4	0	0	8	0	4	5	3	0	3	10	7	5	8	5	0	4	0	0	0	8	4
u_5	0	0	0	0	6	5	7	0	7	0	0	7	8	0	0	0	0	5	0	8	0
u_6	6	0	0	5	8	3	5	8	3	8	0	0	10	0	3	6	0	3	0	8	6
u_7	0	0	9	3	9	0	3	6	1	0	7	6	0	7	0	0	0	3	9	0	2
u_8	0	9	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0
u_9	8	5	9	7	4	1	0	0	1	10	7	3	0	7	3	0	4	1	8	0	0
u_{10}	6	0	9	0	8	0	0	0	0	0	9	0	0	3	0	0	0	0	0	0	0
u_{11}	10	0	0	0	6	7	3	6	0	0	0	7	10	0	0	8	9	0	0	0	8
u_{12}	10	8	8	5	0	7	9	6	0	10	0	5	0	0	9	0	6	5	6	0	0
u_{13}	0	3	9	0	8	0	3	0	5	10	0	0	0	0	8	0	0	0	0	0	0
u_{14}	6	7	9	0	0	5	0	0	0	9	9	0	8	0	0	4	7	0	9	0	0
u_{15}	0	8	0	0	9	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	6
u_{16}	0	0	0	0	0	7	0	8	0	0	0	0	0	9	0	0	0	0	0	0	0
u_{17}	6	7	9	0	9	0	0	6	6	8	7	0	10	4	4	4	2	3	8	8	0
u_{18}	0	9	0	0	0	7	0	0	0	0	0	0	10	0	8	4	6	0	9	8	4
u_{19} (A_1)	0	6	7	0	10	4	5	0	0	0	0	0	10	0	0	0	0	10	0	0	0
u_{20} (A_2)	0	7	8	0	10	5	7	0	0	0	0	0	10	0	0	0	0	10	0	0	0
u_t	6	1	7	0	8	0	5	4	0	10	0	0	8	0	0	4	-	-	-	-	-

Table 3.2 Item Profit Array

Item	Profit	Item	Profit	Item	Profit
i_1	36	i_8	58	i_{15}	64
i_2	38	i_9	65	i_{16}	110
i_3	70	i_{10}	26	i_{17}	99
i_4	123	i_{11}	23	i_{18}	87
i_5	195	i_{12}	93	i_{19}	56
i_6	151	i_{13}	35	i_{20}	33
i_7	111	i_{14}	65	i_{21}	73

Table 3.3 Pre-processed information in the database

User	$w(t,k)$	s_k	$\sum_{j=l}^{j=n} r_{k,j} * p_j$	$w(t,k) * \sum_{j=l}^{j=n} r_{k,j} * p_j$	User	$w(t,k)$	s_k	$\sum_{j=l}^{j=n} r_{k,j} * p_j$	$w(t,k) * \sum_{j=l}^{j=n} r_{k,j} * p_j$
u_1	0.76	0	1949	1481.24	u_{11}	0.33	0	2147	708.51
u_2	0.84	0	429	360.36	u_{12}	0.55	0	792	435.6
u_3	0.10	0	688	68.8	u_{13}	0.88	0	264	232.32
u_4	0.77	0	1763	1357.51	u_{14}	0.59	0	988	582.92
u_5	0.00	0	1389	0	u_{15}	0.87	0.49	870	756.9
u_6	0.56	0	1407	787.92	u_{16}	0	0	504	0
u_7	0.76	0	986	749.36	u_{17}	0.60	0	1518	910.8
u_8	0	0	0	0	u_{18}	0.24	0	1426	342.24
u_9	0.51	0	788	401.88	u_{19}	0.72	0.17	870	635.1
u_{10}	0.65	0	168	109.2	u_{20}	0.80	1	870	700.16

In this example, we have 18 users (u_1 ~ u_{18}) with 21 items (i_1~ i_{21}). The rating range is from 1 to 10. The shilling attacker level τ is kept at 0.5. The attackers' profiles are simulated based on the bandwagon attack model with AFM. The filler size and the attack size are both set as 10%. Hence, two attackers' (A_1 and A_2) profiles are generated and their target items is i_{18}.

Step 1: Estimate the attacker probability for each user (s_k) by following steps shown in section 3.2.1.

Step 1.1: select the cluster with the highest GRMDA, which containing user $(u_5, u_{15}, u_{19}, u_{20})$.

Step 1.2: calculate the average of all of the distances between each selected user u_k and the other users in the g-dimensional space. The results are: $dist_5 = 0.634$, $dist_{15} = 0.398$, $dist_{19} = 0.432$ and $dist_{20} = 0.361$.

Step 1.3: calculate s_k through (3). The output is shown in Table 3.3.

Step 2: Retrieve the target customer' (u_t) profile (shown in Table 3.1) and calculate the similarity matrix between u_t and u_1~u_{18}, as shown in Table 3.3. In this example, the predicted items for u_t are i_{17} ~i_{21}.

Step 3: Calculate $\sum_{j=l}^{j=n} r_{k,j} * p_j$ for k = 1~20 and the results are listed in Table 3.3.

Therefore, the problem is constructed as:

Max:

1481.24*x_1+360.36*x_2+68.8*x_3+1357.51*x_4+0*x_5+787.92*x_6+749.36*x_7+0*x_8+401.88*x_9+109.2*x_{10}+708.51*x_{11}+435.6*x_{12}+232.32*x_{13}+582.92*x_{14}+756.9*x_{15}+0*x_{16}+910.8*x_{17}+342.24*x_{18}+635.1*x_{19};

s.t. 0.49*x_{15}+0.17*x_{19} <0.5;

$\sum_{k=1}^{k=20} x_k > 1$;

Neighbor selection results: u_{20} is excluded from neighbors in any cases and u_{15} and u_{19} can not be selected simultaneously. Since selecting u_{15} has a higher expected profit for e-retailer, u_{15} is kept. In addition, if the e-retailer does not have s specific level of the expected profit, the neighbor selection results would be u_1, u_4, u_{17}, u_6, u_{15}, u_7, u_{11}, u_{14},

$u_{12}, u_9, u_2, u_{18}, u_{13,}\ u_{10}$ and u_3. The recommendation for u_t could be generated based on the selected neighbors.

3.4. Experimental Results

In this section we evaluate the performance of the proposed VNS method. We use a Book Crossing database from GroupLens Research (http://www.grouplens.org/), consisting of approximately 1,149,780 ratings (explicitly and implicitly) provided by 278,858 users for 271,379 books. Since we focus on recommendation based on explicit rating in this study, a subset of 185,973 books, 77,805 users, and 433,671 ratings is considered.

In order to address data sparsity problem, data reduction is a necessary approach. The original subset is reduced according to the number of rating for each user (no less than 20) and the rated frequency of each book (no less than 50). After pre-processing, we remove rating records from books that are rated infrequently or those from users who having rated relatively few books. In addition, the data set provides the ISBN of each book as well as other related information. We have collected the post price for those books from Amazon.com based on the same book format. Due to the confidentiality, we are not able to gather the actual cost of books. Therefore, we estimate the cost and profit by following two previous studies (Jiang et al. 2011, Sampson 2007) and adopt the information distribution U (0.60, 0.80) of the posted price. The books without any price information are also removed in this study. Finally, a subset of 1836 books, 866 users, and 12619 ratings is obtained. The ratings were expressed in a scale from 1 to 10.

In order to examine the performance of the proposed method, we prepare two sets of users: testing and training. We randomly select 20% of the users into the test set and the remaining 80% of the users into the training set. Therefore, neighbors for all users in the test set will be selected from users in the training set. We also withhold 20% of the book ratings as predicted items for the test users to make prediction and the remaining 80% of the ratings are used to calculate the similarity between users in two sets.

3.4.1. Accuracy-Based Performance Evaluation

To validate the performance of the proposed method, we compare our VNS approach with different benchmarks. Since the VNS intends to maximize profits while detecting attackers, we compare the VNS with both attacker detection model and profit-driven model. First, we compare the VNS with the Pearson Correlation Coefficient (PCC) and shilling attack detection (SD) method by Lee and Zhu (2012). In the experiments, we first employ VNS and SD to select proper users and then generate ratings for the predicted items. For PCC, we directly generate ratings based on all users in the training set. Thus, in order to compare the effectiveness of different methods, we adopt MSE to compare the predicted rating and the actual ones, which is given by:

$$MSE = \frac{1}{\sum_{k \in U_T} \|I(k)\|} \sum_{k \in U_T} \sum_{j \in I(k)} \left(r_{k,j} - \hat{r}_{k,j}\right)^2 \qquad (7)$$

where $r_{k,j}$ is the actual rating of user k on item j, $\hat{r}_{i,j}$ is the corresponding predicted rating by the recommender method, $I(k)$ is the set of items that the user k has rated and U_T is all the users in the test group.

For training set, we simulate attackers based on the random attack model, the average attack model and the bandwagon attack model with AFM. For each type of attack model, we vary the attack size {3%, 5%, 10%} and the filler size {5%, 8%, 15%, 25%, 40%, 60%, 85%}. The shilling attacker level threshold is kept at 0.5. Only push attacks are simulated in this study.

For all methods in comparison, the profit is evaluated by total expected profit (TEP):

$$TEP = \sum_{k \in U_T} \sum_{j \in I(k)} \hat{r}_{k,j} * p_j \quad (8)$$

where U_T is all the users in the test group, and $I(k)$ is the set of rated or selected items.

The experimental results are summarized in Tables 3.4 and Table 3.5. In Table 3.4, the comparative results corresponding to the MSE between our proposed method and two benchmark methods (PCC and SD) at seven different filler sizes are reported in seven columns respectively. In Table 3.5, the comparative results corresponding to the expected profit between our proposed method and two benchmark methods (PCC and SD) at seven different filler sizes are reported respectively as well. From Table 3.4, we can see that if the attack model is random attack or average attack, the MSE of our proposed method has the obvious advantage, in particular when the filler size or the attack size is small. Another interesting finding is that if the attack mode is random or average, and the filler size or attack size is low, the MSE of the SD, which is the effective detection method developed in previous study, can be larger than that of the PCC, which does not discriminate shilling attackers at all. It is since in such environment, SD suffers both high false positive rate and

low true positive rate. Despite that PCC does not filter out any attackers, all genuine users are kept in the systems so that it would not lose any information. For the VNS method, since we provide the probability rather than a binary decision for suspicious users, the information they containing can be reserved. This finding further confirms the necessarily of using the discounting-based strategy instead of the filtering-based strategy.

Although SD is very effective when the attack model is bandwagon, we can see that the MSE of our proposed VNS method is still slightly smaller than those from SD and consistently smaller than those from PCC. In addition, from Table 3.5, we can see that our proposed VNS method yields consistently larger expected profits than those from SD and significantly larger than those from PCC. In general, the experimental results show that our proposed method maximizes profits while maintaining the prediction accuracy very well compared to the benchmarks.

Since SD has a relatively better performance in the bandwagon attack model, we further examine the impacts of attack and filler size on methods performance for the bandwagon attack model. The results are shown in Figure 3.2 and Figure 3.3. As we can see from Figure 3.2, as the filler size decreases, our proposed method performs better (i.e. generates smaller errors in comparison with SD and significantly better than those from PCC). In a similar fashion as we can see from Figure 3.3, as the filler size becomes smaller and smaller, our proposed method produces larger expected profits than those from SD and PCC. Overall, the advantage of VNS to SD on both accuracy and profit is especially large when the filler size is small.

Table 3.4 Comparative Results for MSE

Attack Type	Attack Size	Method	Filler Size						
			5%	8%	15%	25%	40%	60%	85%
Random Attack	3%	VNS	0.6226	0.6192	0.6107	0.615	0.6242	0.6243	0.6243
		PCC	0.623	0.6235	0.6322	0.6356	0.6387	0.651	0.6523
		SD	0.832	0.7235	0.6302	0.6243	0.6243	0.6243	0.6257
	5%	VNS	0.6232	0.6203	0.6132	0.6155	0.6243	0.6243	0.6243
		PCC	0.6243	0.6256	0.6328	0.6363	0.6392	0.6525	0.6551
		SD	0.7636	0.7326	0.6245	0.6256	0.6243	0.6243	0.6243
	10%	VNS	0.6235	0.6216	0.6168	0.6172	0.6243	0.6243	0.6243
		PCC	0.6264	0.6277	0.6255	0.6378	0.6431	0.6533	0.6589
		SD	0.7325	0.7129	0.6259	0.6243	0.6243	0.6243	0.6243
Average Attack	3%	VNS	0.6228	0.6205	0.6145	0.6152	0.6241	0.6243	0.6243
		PCC	0.6235	0.6238	0.632	0.6362	0.6428	0.6556	0.6612
		SD	0.6936	0.6266	0.6256	0.6258	0.6243	0.6243	0.6243
	5%	VNS	0.6236	0.621	0.6169	0.6185	0.6243	0.6243	0.6243
		PCC	0.6252	0.6266	0.6322	0.6368	0.6432	0.6562	0.6621
		SD	0.6258	0.6256	0.6255	0.6252	0.6243	0.6243	0.6243
	10%	VNS	0.6239	0.6222	0.6172	0.619	0.6243	0.6243	0.6243
		PCC	0.6282	0.6289	0.6361	0.6392	0.6477	0.6569	0.6696
		SD	0.6256	0.6252	0.625	0.6243	0.6243	0.6243	0.6243
Bandwagon Attack (AFM)	3%	VNS	0.6258	0.6249	0.6196	0.6193	0.6203	0.624	0.6243
		PCC	0.626	0.6269	0.6333	0.6408	0.6396	0.6543	0.6543
		SD	0.6259	0.6253	0.625	0.6248	0.6243	0.6243	0.6243
	5%	VNS	0.6251	0.6247	0.6192	0.6217	0.6242	0.6243	0.6243
		PCC	0.6271	0.6285	0.6324	0.638	0.6464	0.6571	0.6663
		SD	0.6252	0.625	0.6247	0.6246	0.6243	0.6243	0.6243
	10%	VNS	0.6243	0.6241	0.6236	0.6226	0.6243	0.6243	0.6243
		PCC	0.6295	0.6327	0.6481	0.6485	0.6492	0.6582	0.6781
		SD	0.6251	0.6247	0.6246	0.6243	0.6243	0.6243	0.6243

Note. Cell Value is Prediction Accuracy based on MSE

Table 3.5 Comparative Results for TEP

Attack Type	Attack Size	Method	Filler Size						
			5%	8%	15%	25%	40%	60%	85%
Random Attack	3%	VNS	2355	2369	5528	5682	4520	2490	2490
		PCC	1915	2086	2056	2220	2035	1881	1963
		SD	1888	2211	2135	2228	2490	2490	2490
	5%	VNS	2463	3123	4835	5523	3974	2490	2490
		PCC	1931	1802	1772	2012	1692	1982	1882
		SD	2082	2192	2212	2306	2490	2490	2490
	10%	VNS	2892	3321	3183	4231	3981	2490	2490
		PCC	1710	1821	2184	2362	2188	1836	1928
		SD	2382	2186	2358	2490	2490	2490	2490
Average Attack	3%	VNS	2390	2682	5210	5370	4282	2490	2490
		PCC	1820	1932	2026	2216	1872	1966	1798
		SD	2320	2298	2388	2420	2490	2490	2490
	5%	VNS	2328	2688	4382	3890	2490	2490	2490
		PCC	2026	2036	1828	1956	1938	1898	1828
		SD	2156	2278	2430	2456	2490	2490	2490
	10%	VNS	2396	2796	4188	3992	2490	2490	2490
		PCC	1628	1722	1838	2026	2236	1936	1826
		SD	2312	2368	2462	2490	2490	2490	2490
Bandwagon Attack (AFM)	3%	VNS	2360	2350	5470	5630	4860	3220	2490
		PCC	1912	1915	2013	1820	1792	1586	1586
		SD	2069	2100	2150	2280	2490	2490	2490
	5%	VNS	2310	2370	5662	4280	2530	2490	2490
		PCC	1810	1970	2480	1850	1610	1550	1630
		SD	2080	2150	2320	2430	2490	2490	2490
	10%	VNS	2338	2880	3260	2980	2490	2490	2490
		PCC	1730	1990	1635	1782	1629	1582	1562
		SD	2380	2320	2510	2490	2490	2490	2490

Note. Cell Value is Expected Profit

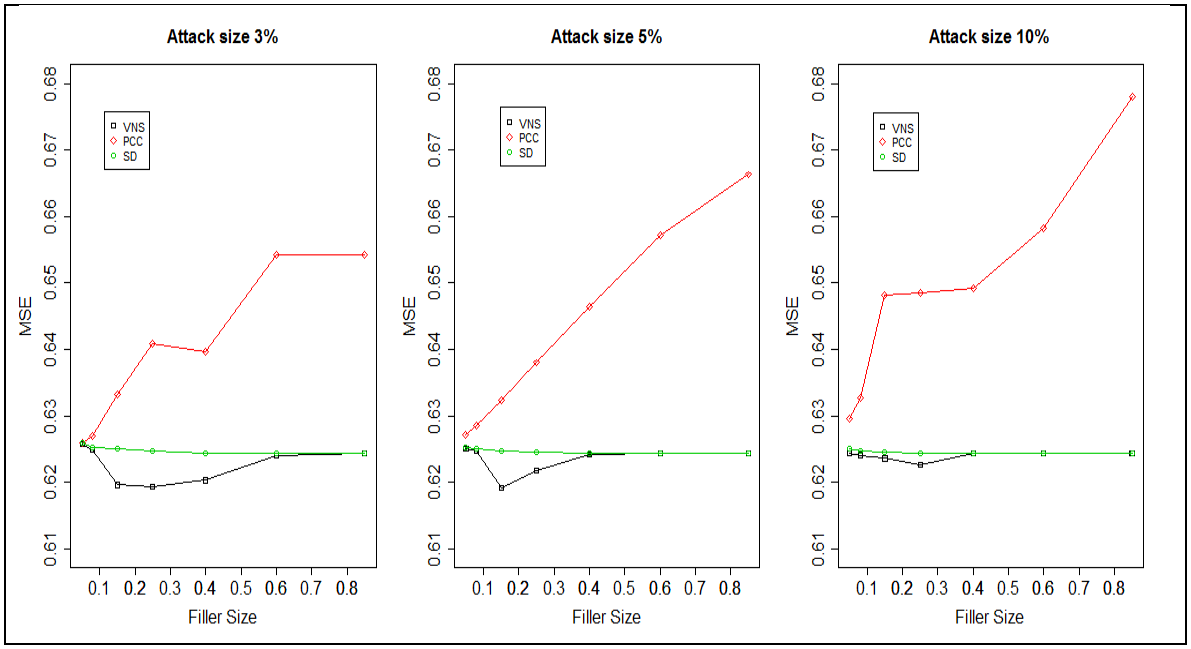

Figure 3.2 Impact of Filler Size and Attack Size on Prediction Error

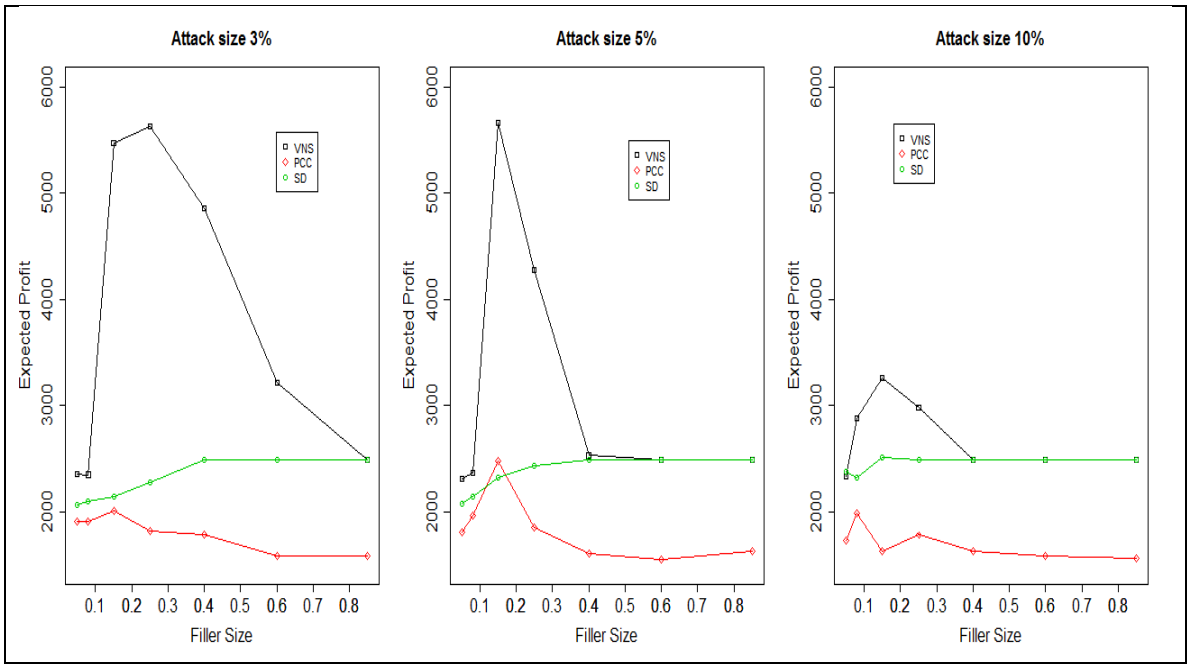

Figure 3.3 Impact of Filler Size and Attack Size on Expected Profit

3.4.2. Value-Based Performance Evaluation

Second, we compare the VNS with the HPRS brought by Chen et al. (2008), which is the pure profit-driven method. Since the HPRS model does not consider the effect of attackers, its vulnerability on attacks is weak. Thus, we compare our method with an integrated approach (SD+HPRS), which firstly uses SD to remove attackers and then recommends the similarity-based profitable product. Since SD is more effective under the bandwagon attack model, we simulate attackers only based on the bandwagon attack model with AFM. Still, we vary the attack size {3%, 5%, 10%} and the filler size {5%, 8%, 15%, 25%, 40%, 85%}.

Since the output of HPRS is a list of recommended items, after generating the predicted rating for each predicted item, we select the top N items with the highest rating. In the testing dataset, items are categorized as "like" if the actual rating is 10 and "dislike" if the actual rating is below 10. Also, we choose precision, recall and F1 to compare performance, as shown in (9), (10) and (11).

$$Precision = \frac{True\ positive}{True\ positive + False\ positive}, \tag{9}$$

$$Recall = \frac{True\ positive}{True\ positive + False\ negative}, \tag{10}$$

$$F\_measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}, \tag{11}$$

Table 3.6 Comparative Results between VNS and (SD+HPRS) method

	VNS				SD+HPRS			
(Attack%, Filler%)	Precision	Recall	F1	TEP	Precision	Recall	F1	TEP
(3%, 5%)	0.682	0.569	0.6203	137	0.359	0.280	0.314	140
(3%, 8%)	0.679	0.566	0.6174	137	0.358	0.280	0.315	139
(3%, 15%)	0.681	0.567	0.6188	126	0.358	0.281	0.315	138
(3%, 25%)	0.688	0.570	0.6235	126	0.358	0.280	0.315	139
(3%, 40%)	0.681	0.566	0.6182	130	0.362	0.282	0.317	139
(3%, 85%)	0.684	0.570	0.6218	137	0.358	0.280	0.315	139
(5%, 5%)	0.68	0.583	0.6278	137	0.357	0.280	0.314	140
(5%, 8%)	0.671	0.560	0.6105	141	0.359	0.279	0.314	140
(5%, 15%)	0.683	0.568	0.6202	137	0.359	0.281	0.315	139
(5%, 25%)	0.681	0.567	0.6188	137	0.358	0.281	0.315	138
(5%, 40%)	0.671	0.558	0.6093	130	0.364	0.285	0.319	139
(5%, 85%)	0.68	0.565	0.6172	137	0.364	0.285	0.319	139
(10%, 5%)	0.681	0.576	0.6241	137	0.358	0.280	0.315	140
(10%, 8%)	0.685	0.566	0.6198	139	0.359	0.281	0.315	140
(10%, 15%)	0.682	0.568	0.6198	137	0.358	0.281	0.315	138
(10%, 25%)	0.684	0.571	0.6224	137	0.364	0.285	0.319	139
(10%, 40%)	0.676	0.567	0.6167	130	0.364	0.285	0.319	139
(10%, 85%)	0.682	0.578	0.6257	137	0.364	0.285	0.319	139

Table 3.6 displays the comparison between VNS and the integrated SD+HPRS approach. The later approach selects items based on the similarity based expected profit.

Despite its advantage in profit, we can see that the TEP generated by VNS is comparable, if not better. From the accuracy perspective, though it adopts SD to overcome the influence from attackers, VNS has a significant advantage on all three measures over those from (SD+HPRS). Hereby, our proposed method achieves a balance between users' expected satisfaction and e-retailer's profit.

3.4.3. Performance Discussion

In this subsection, we discuss the possible underlying reasons behinds the advantage of our proposed VNS over our benchmarks on both accuracy and profit is especially large when the filler size is small.

We evaluate the detection techniques by artificially adding simulated attackers into the original database. When the filler size is small, there is a possibility for SD method to misclassify a genuine user as an attacker. And for SD, each user has a binary decision. The user will be removed once recognized as an attacker. If information provided by this user is valuable, the prediction error will increase. Differently, for VNS, each user will be assigned an attack probability and he/she will be removed in one of the two situations: 1) the attacker probability is very high, which indicates that this genuine user has a high similarity with the simulated attackers compared to other genuine users. In this case, this genuine user, though not a simulated attacker, is a "natural" attacker existing in the original database. Hereby, identifying it as an attacker is actually not a misclassification. Removing it from the database can increase the prediction accuracy; 2) the expected profit from the recommendations generated by this user is very low. In this case, removing this genuine user may increase the profit. Therefore, removing a user does not harm the performance of VNS.

With the increases of the filler size, the performance difference between VNS and SD narrows. It is since when filler size increases (i.e. filler size>40%), SD is more easily to remove all attackers as they have a greater chance of being dissimilar to other users. Thus, when the filler size is large, the selected users by SD are the entire training set without simulated attackers. For VNS, based on the discussion in Section 3.3.1, user k will be assigned a higher attack probability if dist_k is closer to $\text{dist}_{\min}$. When filler size increases, all simulated attackers tend to be gathered at the center of all users (Lee and Zhu 2012). The distance between all attackers with any other genuine users will be shortened and the distance among attackers are close to zero. Hence, $\text{dist}_{\min}$ is more likely to belong to one simulated attacker and $\text{dist}_{\text{mean}}$, $\text{dist}_{\min}$will decrease significantly.

Other simulated attackers are also expected to have a smaller dist_k and a higher attacker probability. But for a genuine user g, the distance from the attackers will be shortened but that with other genuine users will not vary. Since the attack size is small, the distance with other genuine users is much larger than that from the attackers. Hereby, while dist_g may decrease slightly, it is further away from $\text{dist}_{\min}$and even may surpass $\text{dist}_{\text{mean}}$. Accordingly, its attack probability will decrease or be assigned as a zero. This trend is also true for the "natural" attacker. The increasing filler size does not affect his similarity with other genuine users, which takes the majority of the overall distance. Its attacker probability decrease as well. Thus, the genuine users are not likely to be removed as attackers. For VNS, its selected users for recommendations are the entire training set without simulated attackers

as well. This is why the two methods share similar results, which are consistent with those shown in Figure 3.2 and Figure 3.3.

The advantage of VNS to PCC is as expected since PCC in itself, is neither attackers focused nor profit focused. For PCC, its prediction error increases when either filler size or attack size increases since more misleading information is injected by attackers. Thus, the expected profit for e-retailers is affected.

3.5. Discussion and Future Research

Recommendation systems have been shown to be effective for customer retention in various domains. However, retaining customers, while positively related to, is not always equal to maximizing the e-retailer's profit. Even more, the systems are vulnerable to attacks. In this paper, we address these important issues. For collaborative filtering recommendation systems, its success relies on the rating of neighbors who share the same preference with the active user. Hereby, we design a new method to select proper neighbors. We consider the profitability of recommendations to increase the expected gain for the e-retailer while protecting the system from shilling attacks. To validate the performance of the proposed model, we conducted a number of experiments and compared our approach with several benchmarks. Results indicated that the method we proposed in this study could yield better overall accuracy and higher profits. This is particularly true when the filler size is small. Hence, from the customers' perspective, the attacks can be detected and removed so that the recommendations are closer to users' actual tastes. From the e-retailers' perspective, our

approach can lead to significant financial revenue for them. This study suggests the necessity of adopting discounting based strategy in shilling attack detection.

Future studies may also examine the cost and benefit of increasing the attack size when implementing the proposed method. Also, we plan to analytically evaluate how the selection of shilling attack level affects the performance. Further, we did not address the nuke attack in this study. While the advantage of the proposed method is expected to be valid, we intend to simulate nuke attacks in the future study to explore if any pattern exists in the prediction error and profit gain. Moreover, in this study, attackers are simulated by one attack model at one time. In the future study, a mixture of attack models is expected to be applied to simulate attackers, which is much closer to the scenarios in a real electronic marketplace.

CHAPTER 4. DESIGNING INTELLIGENT RATING SYSTEMS WITH RATING FRAUD DETECTION

4.1. Introduction

We are more and more reliant on Internet. For example, Amazon had 209 million users by July 2013(Smith, 2014). By December 2013, 241 million global users were connected in Twitter to share information (Goel, 2014). With advances in information technology, the cyber world has transformed itself as the dominant platform for people to express themselves and connect with others in many parts of their daily lives. Different from the real-world, the interaction in the cyber world is featured by anonymity, in that it can occur between people who do not know each other's real identity. Thus, Internet breaks the geographical limitation and provides a vast collection of information sources.

Despite the convenience resulting from social media for information exchange, interaction with strangers always involves risks. For instance, believing in the rumors spreading across the Internet could lead financial investors to make incorrect decisions. Buyers purchasing products from unreliable sellers may result in severe losses. Hence, people should interact with strangers cautiously to make use of opportunities as well as protect themselves. In the real-world, people would like to interact with entities (e.g. people, items, service, organizations and etc.) that have higher evaluations. Similarly, evaluation can be a critical precautionary measurement for people to regulate their interaction with strangers in the cyber world. In the cyber world, the opinion about an entity is difficult to collect by direct inquiring. Accordingly, rating systems have been designed to help people to judge the quality of strangers beforehand.

Rating systems can collect, calculate, and disseminate evaluations about entities' past behavior. The aggregated feedbacks are rating scores. In most commercial systems, feedback is contributed by users in the form of numerical ratings. In this paper, we term the user providing ratings as *rater*. For example, raters in Amazon can rate the entity in the scale from 1-5 stars, with the higher value indicating the better satisfaction. The rating score is calculated based on the rating from every user and will be updated with the arrival of the new rating. The calculated score is disseminated to all customers as their decision reference. Previous research has already shown that rating system is an effective mean to decrease transaction risks, facilitate buyer satisfaction, and generate premiums for e-retailers (Ba and Pavlou 2002; Houser and Wooders 2006).

In spite of its effectiveness, rating systems are vulnerable to raters' manipulation. Raters may inject biased evaluation for entities to exploit their own benefits. We term the behavior that the rater will provide unfair ratings as *rating fraud* and such users are *fraudulent raters*. There are rating management organizations that provide professional services for online rating manipulation. For instance, nineteen review management companies were caught and fined $350,000 for injecting faking consumer ratings in various sites including Yelp, Google Local, and Yahoo Local in early 2014(Sved 2014). As a result, the rating score is biased by such organized and profit-driven activities. It will undermine the trustworthiness of the rating systems and users' satisfaction will be lost. Therefore, it is necessary to develop mechanisms against rating fraud.

While a growing body of research has developed solutions for rating fraud, they suffer several limitations that are not fully addressed. In one way, the detection of the suspicious rater relies on the examination of his or her rating deviation from the majority rating or the earlier ratings (Fei et al. 2013; Jindal and Liu 2008; Lim et al. 2010; Liu and Sun 2010; Mukherjee et al. 2013). This will generate misleading results when the majority or the early raters are fraudulent. In addition, the normal users' rating values for a specific entity can have a large variance due to the subjective difference. In another way, it first identifies a reliable rater and utilizes his or her rating experience to filter out the suspicious raters by comparing similarity (Dellarocas 2000; Teacy et al. 2006). However, when such a reliable rating is lacking or when there are newcomers, the application of this type of method faces difficulty. In this study, we intend to propose the method which detects the fraudulent raters based on the features of the rating series rather than its deviation of values. We first identify the suspicious entity by fitting its rating series to the autoregressive moving average model. If the model has a good fit, this entity is selected as a potential attacked entity since its ratings are not independent. Then we retrieve a list of users who have rated this entity. For users in this short list, we adopt clustering-based methods to discriminate fraudulent raters based on their rated entities and rating timestamps.

The proposed method addresses the fraudulent raters in the collaborative rating fraud. While a singleton can conduct a personal attack independently, the influential rating fraud is usually organized and planned. The collaborative fraud occurs when a seller or the rating management organization can control multiple users or user IDs to inject unfair ratings strategically on the target entity(s). Compared to the individual unfair rating, collaborative

rating fraud brings more significant challenges to the accuracy of the rating systems and thus it is the focus of the present study.

In the next section we provide a brief review of research on rating fraud model, followed by an overview of the literature on rating fraud detection. In Section 4.3, we introduce the proposed two-phase procedure for rating fraud detection. Section 4.4 presents the experimental studies that we conducted to evaluate the method performance. Finally, section 4.5 concludes the papers and discusses the future research directions.

4.2. Literature Review

4.2.1. Rating and Rating Systems

Two types of measurements for rating have been examined in electronic markets, which are non-computational and computational (Zaacharia et al. 2000). Non-computational rating does not provide a numerical value, but rather, records all the activities related to the opinion. A notable example of a non-computational rating system is "Better Business Bureau Online", whose main functions are handling disputes and tracking complaints (Azari et al. 2003). On the other hand, computational rating is calculated based on the evaluations collected from all the evaluators. For computational rating, there are two types of rating systems: content driven and user driven. Content driven systems (such as WikiTrust) use the automated content analysis to derive ratings by comparing contributed content with ground truth. The less frequently the content is modified, the more reliable it is. Content-driven rating systems face several challenges. Due to the automatic calculation algorithms, it may reduce the users' belief of the scores since they do not understand the internal calculation

process. In addition, these systems rely on the reviews. If everyone is lazy, the rating will be misleading. However, due to the visibility of the content modification to all, these systems are more resistant to users' manipulation (Chatterjee et al. 2008). Hereby, this study focuses on the user driven rating.

User driven ratings (such as eBay and Amazon) compute the rating based on explicit user feedback. For user driven rating systems, the rating score can be calculated either as the difference between all positive scores and negative scores (e.g. eBay) (Resnick and Zeckhauser 2002), as the mean of all ratings (e.g. Amazon) (Schneide et al. 2000), or as the weighted average of all the ratings where the weight can be the rating age, the rater helpfulness, etc. (Liu et al. 2013). In the more complex way, for example Beta Reputation Systems (BRS), it utilizes the previous positive and negative ratings as parameters to formulate the beta probability density functions. Based on the previous rating score and the new rating, it can calculate the updated rating score (Jøsang et al. 2007). While it enables the intuitive understanding of the calculation process, user-driven systems are open to malicious users' manipulation. In the next subsection, we will discuss the conventional rating manipulation model.

4.2.2. Rating Fraud Model

Rating fraud can be classified from several dimensions. The first dimension is the proportion of honest raters in the overall environment. In most cases, the majority of raters are honest. However, it is possible that the ratio of the malicious raters is predominant or they control a larger amount of user IDs. This scenario is termed as *Sybil Attack* (Douceur 2002).

The second dimension is the rating value. It can be classified as *Ballot Stuffing* if the unfair high ratings are injected or *Bad Mouthing* if the target entity's rating is undermined (Dellarocas 2000). The third dimension is the rater's manipulation activity. Three types of malicious behaviors have been found (Irissappane et al. 2012).

- *Consistent Attack*: users consistently provide unfair high (low) ratings to entities with low(high) quality;
- *Camouflage Attack:* users camouflage as honest ones by providing fair ratings strategically. For instances, besides unfair rating, they may also inject fair ratings to non-target entity to pretend as a normal user;
- *Whitewashing Attack:* users register new accounts after a period of injecting unfair rating to whitewash their history;

4.2.3. Rating Fraud Defense Mechanisms

Various defense mechanisms have been proposed to deal with rating fraud. They can be classified as preventative or detective mechanisms. Preventative mechanisms intend to discourage fraud by increasing its costs. For instance, by combing user accounts with IP addresses or using social networks to detect nodes with multiple fake identities, it will increase the difficulty for malicious users to control multiple IDs (Douceur 2002; Yu et al. 2006). Epinions encourages raters to provide honest feedbacks by sharing income with them (Jøsang et al. 2007). Such preventative approaches are unable to capture fraudulent activity and they are ineffective when users enhance their hiding techniques (e.g. IP address spoof) or if they can realize more significant profits by injecting attacks.

Another detective solution is detecting suspicious users according to various features. One stream is to classify a user as a fraudulent rater by his or her rating feature. Jindal and Liu (2008) adopt logistical regressions with a multitude of rater characteristics including the ratio of the first product rating, the number of single raters, mean and variance of all his or her ratings. Lim et al. (2010) explore fake rating through predefined types of behavior abnormalities (i.e. extremely high or low ratings). Wang et al. (2011) use the graphical method for rating fraud detection by considering the relationship among raters, ratings and entities. Mukherjee et al. (2013) exploit characteristics of abnormal behaviors and designed author spamicity model for detection. Fei et al. (2013) employ those features in Loopy Belief Propagation (LBP) with several raters' features such as individual rating deviation, ratio of verified purchase to detect fraud. Liu et al. (2013) propose a fuzzy logic which combines user's rating time, rating value similarity and rating quantity to against unfair ratings.

Another stream compares the target user's rating with the overall rating trend from other users to estimate whether it is a fraudulent user. In BRS, if the overall entity rating falls in the rejection areas of the beta distribution of the target user's ratings, this user is considered dishonest due to the *majority rule* (Whitby et al. 2004). However, this method is vulnerable to Sybil Attack. Instead of relying on the majority rule, Liu et al. (2011) propose the method assuming that the entry of a large amount of malicious ratings would lead the sudden change of the overall rating of the entity. Hereby, they discuss how to locate the rating change by comparing new ratings with previous ones. The evaluation of this method demonstrates increasing the accuracy and stability over the beta-function based defense method. However, this model may be vulnerable when the malicious users are the early raters.

Another method TRAVOS is proposed by Teacy et al. (2008). It evaluates the trustworthiness of the target user by comparing its rating of the entity with that of other similar entities. This method assumes that users have a constant behavior, which may not be the case in Camouflage Attack.

4.3. Rating Fraud Detection

In this section, we discuss the proposed method in details. This study addresses the collaborative rating fraud. It can be carried out by multiple users or by one user controlling multiple user IDs. We refer each of these users as a fraudulent rater. Fraudulent raters have their *target entity(s)* in which they inject unfairly high (Ballot Stuffing) or low (Bad mouthing) ratings. They can also provide ratings beyond the target entity (or entities). However, for each entity, every malicious user ID can only inject one rating throughout the entire period. This is due to the fact that the duplicate rating detection has already been widely discussed and is much easily controlled (Jindal and Liu 2008).

Due to the collaborative rating's fraudulent nature, multiple malicious users could be associated with each target entity. Thus, our defense mechanism includes two steps: 1) identify the potential target entity(s) based on the entity features; 2) Retrieve the associated users and filter out the fraudulent raters by user features. Accordingly, we will first discuss two important characteristics related to rating fraud that are useful in supporting our defense methodology. Each characteristic is identified with empirical evidence and detailed analyses. We will use a real-world cyber competition dataset (Liu et al. 2011) to illustrate those characteristics. This dataset includes both normal data and attack data. For each piece of

record, it includes both rating time and rating value. A detailed description of this data set will be provided in Section 4.4.

4.3.1. Target Entity Rating Series

For each entity e, we create a time series for all its ratings over the time.

$$r_t^e = \{ r^e{}_{t_1}, r^e{}_{t_2}, \ldots r^e{}_{t_k} \};$$

Where t_k is the order for this rating instead of its accrual time. For example, $r^e{}_{t_1}$ indicates that it is the first rating for the entity e. We use the order (relative time) instead of the accrual time since our concern is the correlation among ratings rather than the change of entity overall rating.

Users have various backgrounds. Intuitively, their ratings towards an entity should be independent with each other if they are honest (Holmes 1994; Hu et. al 2012; Xie et al. 2012). Accordingly, the rating scores for a non-target entity should be mutually independent and identically distributed with respect to time. For example, Figure 4.1 plots $\{r_t^e\text{-}\mu\}$ over the time for a non-target entity selected from our experimental data, where r_t is the rating the timestamp t and μ is the expectation of all ratings. We can find that the ratings are randomly distributed. However, if the entity is under attack, r_t^e may be correlated with each other due to the existence of the malicious users. In the collaborative rating fraud, fraudulent raters are organized together so that their ratings are not independent any more. For instance, Figure 4.2 plots the rating time series for one target entity under three different attack sizes from the experimental data. One notable finding we can observe from the figure is that the ratings

from fraudulent raters are correlated. Hereby, an autoregressive moving average (ARMA) model is approximate for the target entity rating distribution, which is shown as below.

$$(r_t^e - \mu) = \sum_{i=1}^{p} \varphi_i (r_{t-i}^e - \mu) + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

where ρ, q is the autoregressive and moving average model order respectively, φ_i, θ_i are the parameters of the autoregressive and moving average model respectively and ε_t is the error term.

Despite certain cases in which a self-selection process may exist so that ratings from normal users may not be random (Li and Hitt 2008). However, Hu et al. (2010) analyzed the data from Amazon and Barnes and Noble, and show that regardless of self-selection, rating fraud must exist in the entity if there is dependency among all its ratings. Thus, for the target entity, its rating series is expected to be fitted by the autoregressive moving average model. We can fit the rating series to the ARMA model and examine the model fit. If the model has a perfect fit, it is highly suspected to be the target entity.

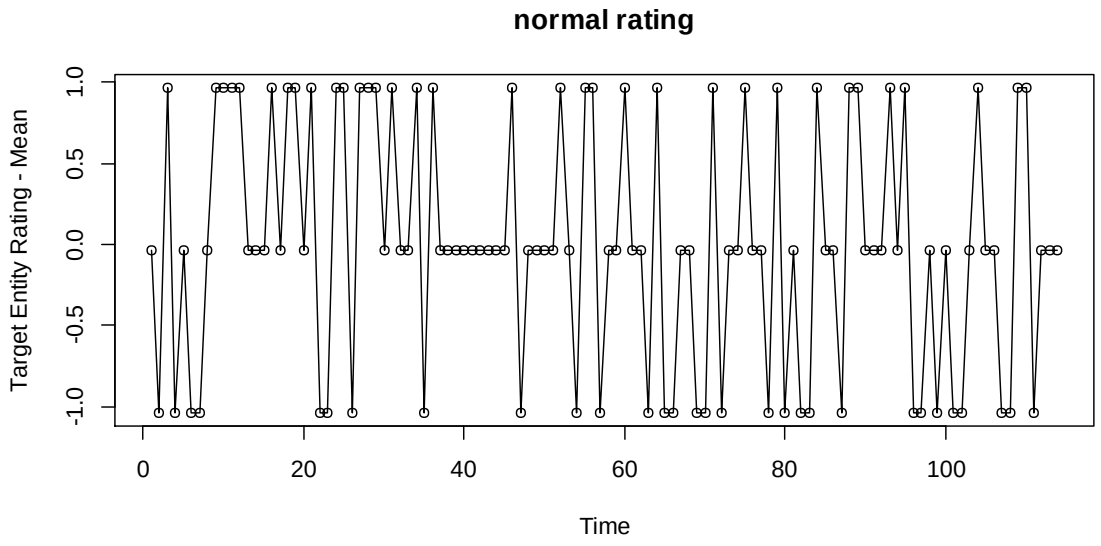

Figure 4.1 Rating Time Series with Genuine Users

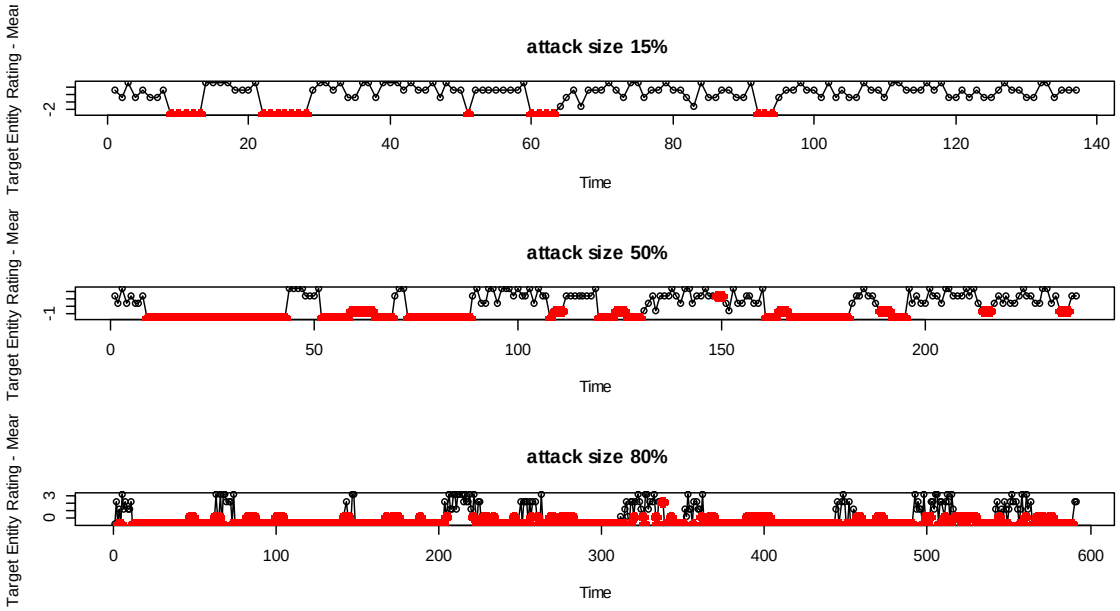

Figure 4.2 Rating Time Series with Fraudulent Raters in Various Sizes

Note. Black circle, genuine users; solid and red circle, fraudulent raters.

4.3.2. Fraudulent Rater Rating Series

Fraudulent raters have fundamentally different objectives from genuine users, as do their behaviors. Genuine users provide ratings for the entity based on their preferences. Individual difference makes it difficult for a group of users to rate exact the same entities. The activities of malicious users are usually controlled and panned, particularly those hired by the organizations. Entities selected to rate by organized malicious users are different from those rated by normal users. For malicious users, there are two types of entities to rate: target and non-target ones. For target entities, they rated exactly the same set since they are preselected and mostly are not interested by genuine users. For non-target ones, malicious raters selected entities randomly in order to camouflage themselves. Despite the randomness, the non-target entities selected by collaborative malicious user IDs would still have a high similarity since several user IDs may be controlled by the same user. The user with several malicious user IDs may just pick up the same set of entities for convenience. Therefore, malicious users controlled by the same user are expected to have a similar rating pattern. This feature can be utilized to cluster malicious users.

In addition to the rating similarity, fraudulent raters also have unique temporal features in their own rating series. Since a deadline is usually given, profit driven raters usually accomplish their tasks within a short period intensively, i.e. right before the deadline (Parker 2011). While a new deadline may be given in later days, the overall time interval between every two consecutive ratings for a malicious user should be small. This implies that if the fraudulent user rates more than one entity, it is highly possible that ratings are injected in quick succession. However, time interval between ratings for genuine users usually is larger since they need time for consideration before making rating decisions.

This phenomenon has been observed in network intrusion and mobile Apps ranking fraud (Soldo et al. 2009; Soldo 2011; Sorrel 2009; Zhu et al. 2013). Table 4.1 also shows an example from our experiments. In the entire dataset, we randomly select five genuine users and two fraudulent raters, and then retrieve all of their ratings. The seven users had rated a total of 16 entities. The cell value t^{ue} is the accrual day that a rating has been given. We can find that compared to the fraudulent raters, genuine users have a more widely ranged rating time range. We capture and define this feature as the mean rating time distance (MRD), which is calculated as (1):

$$MRD_u = \frac{\sum_2^{n_{u-1}} \min[(t_n^u - t_{n-1}^u), (t_{n+1}^u - t_n^u)]}{(n_u - 2)} \quad (1)$$

where n_u is the total number of entities that the user u has rated and t_n^u is the actual day of the n^{th} rating the user u has given for all possible entities. The last column of the Table 4.1 shows the MRD value for every user. It shows that genuine users have much larger values in MRD compared to malicious users. In addition, we can also note that entities rated by malicious users are quite similar.

Table 4.1 An Example Rating Timestamps Distributions

t^{ue}	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}	e_{16}	MRD
u_1			59	26					100	38			24	95			10
u_2				44			125	45	44	77			59				8.25
u_3	124			121	89	144			143		127			150			2.2
u_4				86				92	101	87	101		101		94	92	0.5
u_5	106			121					145		131		108				7.33
$u_6(f_1)$	14	14				14				14		14		14		14	0
$u_7(f_2)$		44				44				44		44					0

4.3.3. Fraudulent Rater Detection Algorithm

Based on the characteristics of the target entity and fraudulent rater discussed in the previous subsection, our algorithm is designed to consist of two phases. In the first phase, we select the suspicious entity based on the entity's rating series and retrieve its associated raters. In the second phase, we examine the rating series of every rater selected in the first step and discriminate a set of fraudulent raters through clustering with the consideration of the mean rating time distance.

Phase 1. Select a set of P target entities and their corresponding Q users.

Input:

$R = r^{ue}$: (n*m) user-entity rating matrix, where $u \in [1, n], e \in [1, m]$;

$T = t^{ue}$: (n*m) user-entity rating time (day) matrix, where $u \in [1, n], e \in [1, m]$;

Procedures:

Step 1. For each entity e, construct its non-void rating vector $\boldsymbol{r}^e = \{r^e{}_1, r^e{}_2, \dots r^e{}_k\}$and its corresponding rating time vector $t^e = \{t^e{}_1, t^e{}_2, \dots t^e{}_k\}$;

Step 2. Sort $\boldsymbol{r}^e$ based on the values in $\boldsymbol{t}^e$ in ascending order such that $\boldsymbol{t}^e = \left\{ t^e{}_{t_1}, \dots t^e{}_{t_k} \right\}$, $where\ t^e{}_1 \leq \cdots \leq t^e{}_k$. Thus, $\boldsymbol{r}^e = \left\{ r^e{}_{t_1}, \dots r^e{}_{t_k} \right\}$ where t_k is the order for the rating $r^e{}_{t_k}$ in the vector $\boldsymbol{r}^e$;

Step 3. Initialize $Flag_e$=0; Estimate the proper ARMA models for $\boldsymbol{r}^e$ via Yule-Walker method;

Step 4. Apply Ljung–Box test to the residuals from the fitted model in Step 3 to detect the goodness of the fit. If there is a model has a good fit, $Flag_e$=1;

Step 5. Repeat Step 1~4 for all entities. Find the P entities with $Flag_e$=1;

Step 6. Find Q users who have ever rated at least one of the P entities.

Output:

$\bar{T} = t^{ue}$: (q*m) user-entity rating time (day) matrix consisting of the selected users in Step 6, where $u \in [1, q], e \in [1, m]$;

$\bar{R} = r^{ue}$: (q*m) user-entity rating matrix consisting of the selected users in Step 6, where $\in [1, q], e \in [1, m]$;

Phase 2. Select a set of fraudulent raters based on the clustering of their rating pattern.

Input:

$\bar{T} = (t^{ue})$ and $\bar{R} = (r^{ue})$: the output from the phase 1;

Procedures:

Step 1. For each user u, calculate its mean rating time distance from $\bar{T}$ as MRD_u;

Step 2. Construct an (Q*Q) user-user distance matrix $\boldsymbol{D} = (d_{ui})$, based on the user-entity matrix $\bar{R}$. The user-user distance is calculated by using (1-Tanimoto coefficient). Tanimoto coefficient is a general form of Jacarrd coefficient, which has shown a clear advantage over other similarity measures in the case of extremely asymmetric distributed or sparse data vectors such as in the rating data (Mild and Reutterer 2003). For user u and user i, $d_{ui} = 1 - \frac{\widehat{\boldsymbol{r}^u} * \widehat{\boldsymbol{r}^i}}{(||\widehat{\boldsymbol{r}^u}||^2 + \left|\left|\widehat{\boldsymbol{r}^i}\right|\right|^2 - \widehat{\boldsymbol{r}^u} * \widehat{\boldsymbol{r}^i})}$, where $\widehat{\boldsymbol{r}^u}$ and $\widehat{\boldsymbol{r}^i}$ denoting the rating vector for their common rated entities respectively;

Step 3. Cluster users into J groups based on their distance using hierarchical clustering method. The distance among clusters is calculated by ward's method;

Step 4. For each cluster C, C=1, 2, …, J, calculate the group average MRD as $GMRD_C$, which is the mean value of every user's MRD in the particular cluster;

Step 5. Select the cluster with the lowest GMRD.

Output:

The group of users selected in Step 5. They are identified as fraudulent raters and will be suggested to be removed from the user list.

4.4. Experimental Results

In this section, we evaluate the performance of our proposed method and present the experimental results. To test how accurately the proposed method can detect the fraudulent rater, we need to rely on test data in which the true category (i.e. either honest or not) of every user is already known. Even though the user-entity rating datasets are available in certain online systems, they seldom have the dishonest users labeled. Thus, we cannot evaluate the accuracy of prediction due to the lack of actual values. The common solution in previous studies is simulating fraudulent raters based on the assumption of their behavior features. However, the simulation could not reflect the realistic situations comprehensively so that the evaluation of accuracy may be affected accordingly.

In this study, we use a cyber-competition data, which includes both normal rating data and attack rating data (Liu et al. 2011). In both normal and attack data, each piece of rating contains the entity ID, the user ID, the rating time and the rating value. The normal data are collected from a real e-commerce site with rating values in the numeral scale 1 to 5. The set contains 5688 user-entity rating records collected over 150 consecutive days from

300 normal users and 300 products (entities), denoted as (u_1,…..u_{300}) and (e_1,…..e_{300}) respectively. Users (u_1,…..u_{300}) are considered as honest users in this study. The attack data are obtained from an attack competition designed for this e-ecommerce site. The goal of the competition participants is downgrading the rating of one target entity (e_1 in this competition). A participant can realize the goal by rating from multiple user IDs. Although every participant has the same goal in this competition, the individual behavior may vary a lot. As discussed in section 4.2.2, a participant may strategically select non-target entities besides the target entity to rate. Or it may camouflage itself by injecting fair ratings via some user IDs. Regardless of their attack strategy, one participant cannot inject more than 100 ratings, nor control more than 28 user IDs. And a user ID can only inject one rating for one entity. All the ratings from a participant are recorded in an attack file. There are a total of 13028 attack files in the attack data.

For every attack file, we measure its attack effectiveness using the *rating shift*, denoted as $\Delta = \bar{r}^e - \bar{r}^{e\prime}$. The $\bar{r}^e$ and $\bar{r}^{e\prime}$ represent the average rating for the target entity before and after the attack, respectively. Since the goal of this competition is to downgrade the target entity, a larger value of Δ indicates a stronger attack and a smaller value indicates the participant's behavior is very similar to the original normal user. For all the attack files, we categorize them based on their own Δ values into three attack level groups: *weak attack group* with Δ value between [0.1, 0.25), *moderate attack group* with Δ value between [0.25, 0.4), and *strong attack group* with Δ value between [0.4, 0.55). There are 1543, 4254 and 7231 files in each group respectively. It is consistent with the purpose of the competition that

the majority of participants are strong attackers. We test the detection performance of our proposed method in each group respectively.

The effectiveness of the proposed method is tested in two cases. In the first case, we analyze its effectiveness using the original attack data. We add and combine the normal data with every attack file and check the fraudulent rater detection result. However, even though attack files are the "real" attack data, the result for the single attack file testing may not be applicable to the real world due to the limited rater pool. As discussed in section 4.2.2, the pool of the raters or ratings could be much larger in the real world, e.g. Sybil Attack. Hereby, in the second case, we demonstrate the robustness of the proposed method in various attack sizes by constructing multiple attack files strategically. The details of this construction process will be discussed in section 4.4.2. To demonstrate the incremental accomplishment of our method, we compare our proposed approach against a well-known benchmark, BRS (Josang and Ismail 2002; Whitby et al. 2004).

4.4.1. Single Attack File Detection Testing

In the single attack file detection testing, normal data is added to every attack file, which creates 13,028 merged test datasets. The user IDs in the attack file are labeled as actual fraudulent raters while those in the normal data are honest raters. One user-entity rating matrix and one user-entity rating time matrix are generated from the ratings in each test dataset. Following the two steps described in section 4.3.3, suspicious entities are first identified and then a group of users are retrieved as the predicted fraudulent raters. The effectiveness of the method is evaluated by comparing the predicted fraudulent raters with the actual ones.

We first assess the accuracy of identifying the target entities, which is the foundation of the following successful fraudulent rater detection. We adopt *recall*, as defined in (2), to measure among all target entities, what is the percentage of the accurately detected target entity. In this competition, there is only one target entity (e_1). Accordingly, for each single attack file testing, the recall value should be either zero or one. Meanwhile, we use *false positive rate* (FPR), as defined in (3), to calculate among all non-target entities, what is the percentage of the mistakenly identified target entity. Since there are a total of 299 non-target entities in this competition, the value of FPR = the number of false positive/299. For each testing file, we calculate a pair of (FPR, recall). Then, in every attack level group, we average all recall values with the same FPR value. Finally, as shown in Figure 4.3, we plot the overall Receiver Operating Characteristic (ROC) curves for the target entity detection in each attack level group, with x and y axes representing FPR and recall respectively. Obviously, the proposed method can accurately detect the target-entity. Even in the weak attack group, where participants mostly pretend to be the normal users, the proposed method can still detect the entity e_1 as the target entity. Another thing we could notice is that stronger groups have more non-target entitles mistakenly identified as target entities, particularly in the strong attack group. It is since their participants may also inject unfair ratings to the non-target entity, which actually make those entities as the "target" ones. Ratings from participants in the weak attack group are mostly close to the normal users (i.e. just the fair rating). Even they are added to the non-target entities, if any, the entity is still not under attack.

$$Recall = \frac{True\ positive}{True\ positive + False\ negative}, \quad (2)$$

$$FPR = \frac{False\ positive}{False\ positive + True\ negative}, \quad (3)$$

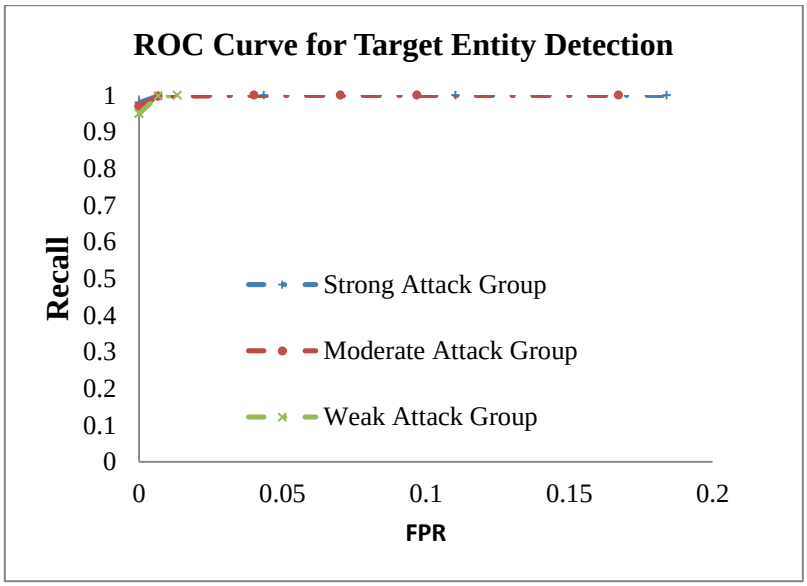

Figure 4.3 Performance of Target Entity Detection

Next we evaluate the fraudulent rater detection performance of the proposed method. In addition to recall, which measures the percentage of the fraudulent raters that are retrieved by the proposed method, we also concern the precision of the detection, which is the ratio of the retrieved raters that are accurately fraudulent, as defined in (4). A lower precision, even with a high recall, indicates a larger number of honest users are probably removed and the rating for the target entity will still be misleading.

$$Precision = \frac{True\ positive}{True\ positive + False\ positive}, \quad (4)$$

Table 4.2 summarizes the overall detection precision and recall of every group. On one hand, the results confirm the effectiveness of the proposed method. We can observe that the group with higher attack level has better performance in both precision and recall perspectives. When the attack level is low, e.g. the weak attack group, the users behave similarly to the normal users so that it is more difficult to differentiate them. Thus, the recall value for the weak attack group is pretty low. When the attack level is high, the proposed method can detect the fraudulent raters accurately. In the strong attack group, the precision and the recall values are above 95%. On the other hand, we can see that the average recall value of the BRS is slightly better than that of the proposed method. Actually, the recall advantage is even larger when the attack level is lower. Its average precision, however, is significantly lower than of the proposed method in average attack level. The precision of the BRS in the weak attack group is below 0.5 (0.42). It indicates that more honest raters than the malicious ones would be removed from the systems, which will hurt the reliability of the systems. The results show that the effectiveness of the proposed method in discriminating the fraudulent raters while leaving the majority of the normal users in the systems.

Table 4.2 The Overall Performance for Single Attack File

	Method	**Weak**	**Moderate**	**Strong**
Precision	Proposed	0.65	0.87	0.98
	BRS	0.42	0.55	0.68
Recall	Proposed	0.41	0.80	0.96
	BRS	0.58	0.82	0.96

A more detailed analysis on the precision and recall in each attack level group for the two methods is presented in Figure 4.4 and Figure 4.5. The precision/the recall result of every testing file, is categorized into five intervals including [0, 0.2], (0.2, 0.4], (0.4, 0.6], (0.6, 0.8], (0.8, 1.0] respectively. For every attack level group, we calculate its own frequency in each interval. Then we plot the frequency distribution of the precision (Figure 4.4 (a) and Figure 4.5 (a)) and the recall (Figure 4.4 (b) and Figure 4.5 (b)) in each group respectively. In both figures, the x-axis represents the precision/recall interval and the y-axis is the relative frequency value in every attack group. Obviously, for the proposed method, almost every testing file in the strong attack group has very high precision and recall values. For the BRS, however, there are still a number of files in the strong attack group has very low precision. For the weak attack group using the both methods, it has 20% testing files with the recall values below 20%, which means only 20% of the fraudulent raters are successfully identified. However, the ratings from the un-identified raters may not be biased largely due to their near to normal behavior. Meanwhile, we observe that the weak attack group using the proposed method has a relatively better performance in precision, with around 2% below 0.2. It indicates that the raters removed are less likely to be the honest users so that the rating quality can retain. Hereby, even though the detection performance in the weak attack group is not high, the rating systems using the proposed method may still maintain its accuracy. However, it is not the case for the BRS method, which has majority low precision values in the weak attack group.

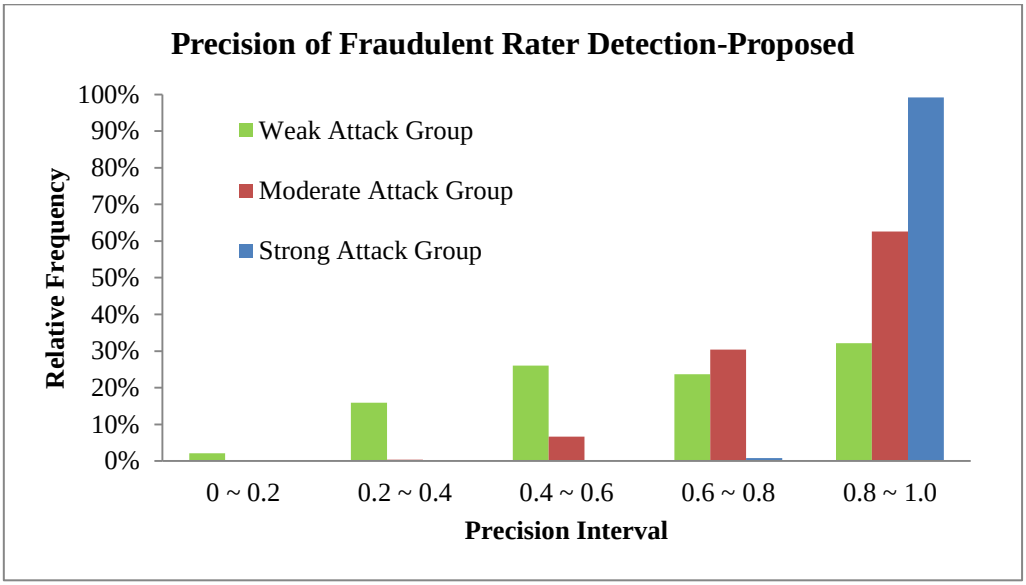

Figure 4.4(a) Precision of Fraudulent Raters Detection for the Proposed Method

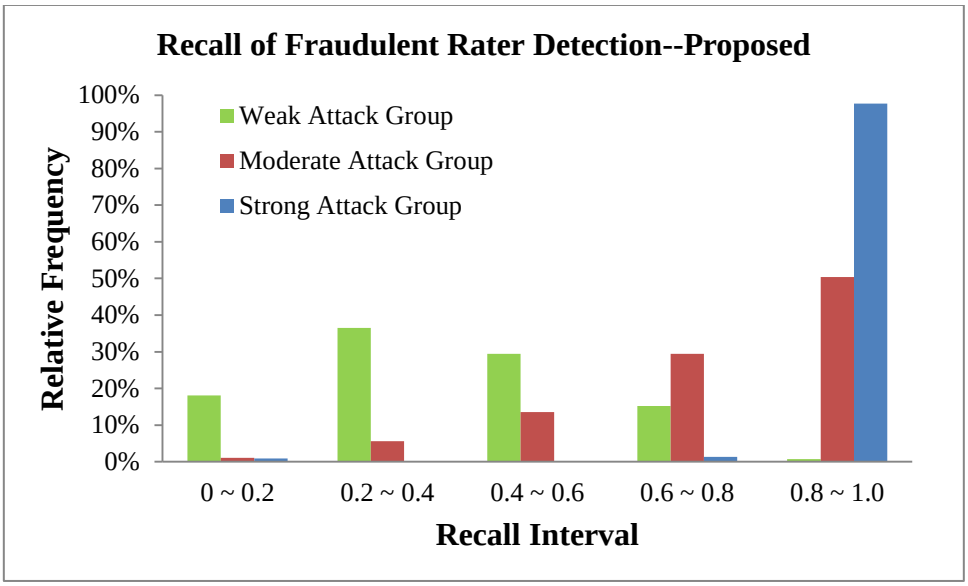

Figure 4.4(b) Recall of Fraudulent Raters Detection for the Proposed Method

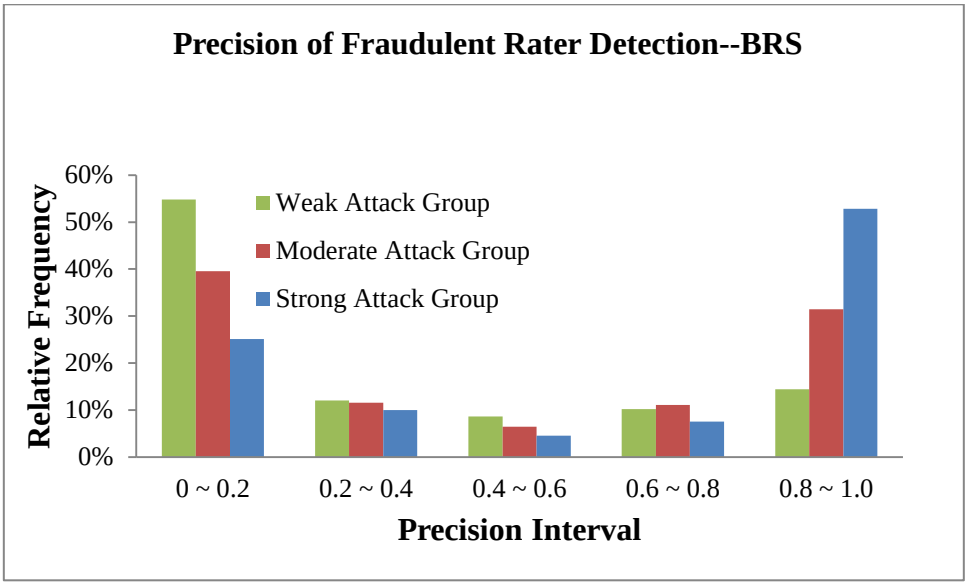

Figure 4.5(a) Precision of Fraudulent Raters Detection for BRS

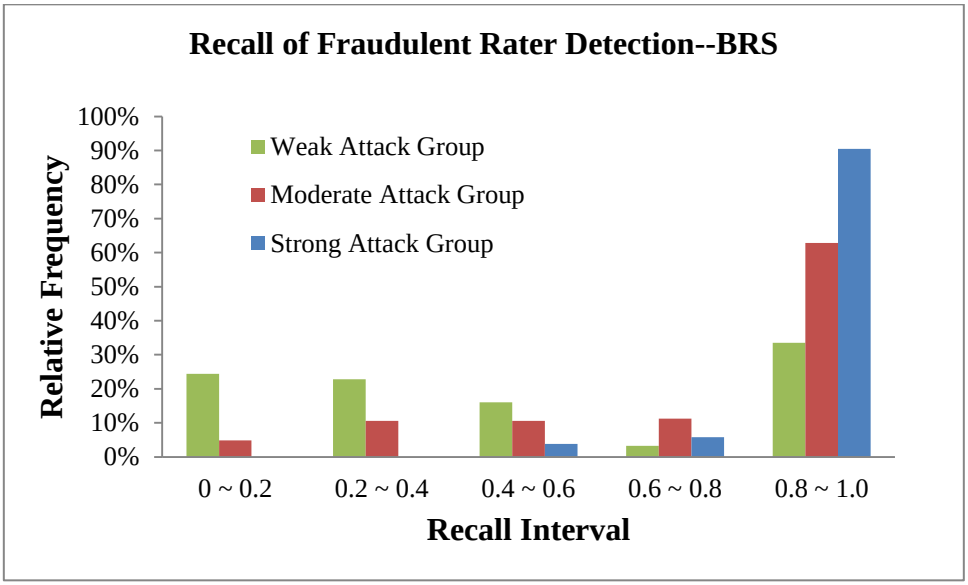

Figure 4.5(b) Recall of Fraudulent Raters Detection for BRS

Based on the above observations, we further assess how the rating of the target entity has been affected after removing all the predicted fraudulent raters by recalculating its rating

shift for every testing file. As introduced earlier, we calculate the difference between the original rating before attack and the current rating after attack detection. The average rating shift for the strong attack group, moderate attack group and weak attack group is 0.0005, 0.0108, and 0.0084 respectively. Compared to the original rating shift interval for each group, the impact from the fraudulent raters is trivial. Figure 4.6 displays the frequency distribution for the rating shift after the detection of every testing file. The x-axis represents three rating shift levels and the y-axis denotes the relative frequency of the rating shift for each group. Clearly, zero is the majority rating shift value in each group after applying the proposed method. For both the moderate and the weak attack group, most of their rating shifts have been significantly eliminated. And in the strong attack group, more than 97% of testing files have returned to the normal rating.

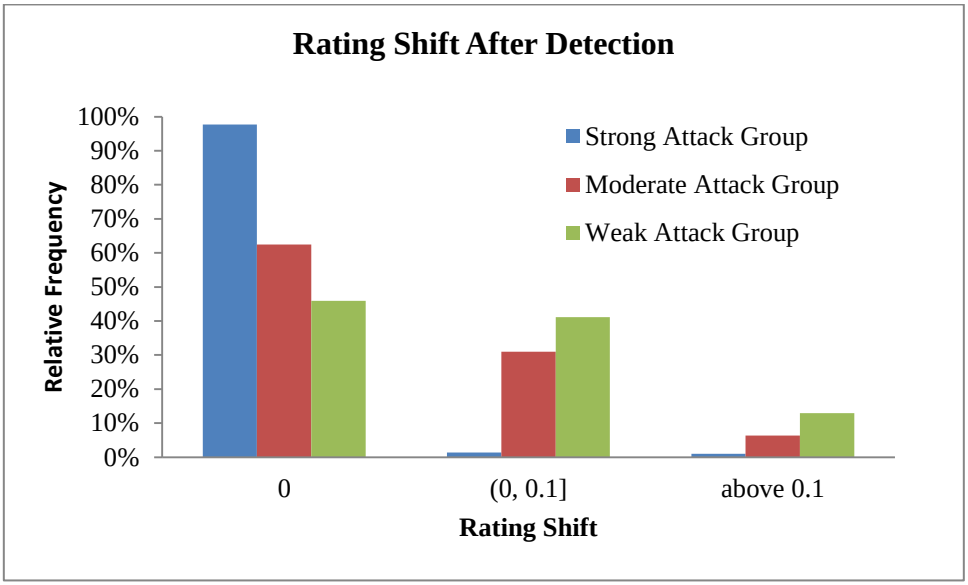

Figure 4.6 Rating Shift after Detection in Various Attack Levels

4.4.2. Multiple Attack Files Detection Testing

Different from the real world, the rater size is limited in the original attack files. We need to create new attack files to represent more diverse attack environments by combining the existing attack files.

We first generate attack files in various attacker sizes. Attacker size is the percentage of the raters who are fraudulent in the attack file. For example, 10% means there are 0.1*p user IDs in the attack file, where *p* is the total number of raters in the rating systems. Nine different attacker sizes are used in that we cover the situation of Sybil Attack. Specifically, we change the attacker size from 10% to 100% in intervals of 10%, and record the changes in performance in terms of precision and recall.

We construct attack files for every attack size. When an attack size is selected, the corresponding number of the fraudulent raters is fixed. For example, suppose the number of the raters is *M*. In each attack group of the original attack data (i.e. weak, moderate and strong), we randomly select a certain amount of attack files which have a total of *m* fraudulent raters. We adjust the user ID in different attack files to make sure no duplicates. Then the selected attack files are merged together in each attack group respectively. After applying the proposed method, the precision and the recall of the fraudulent rater detection is recorded in all three attack groups. For each attack size, we repeat the experiments 500 times and make sure that there is no same combination of the files. Figure 4.7 and Figure 4.8 show the mean values of precision and recall achieved from the 500 experiments for the both methods. Obviously, the impact of the attack size on the performance for the two methods is significantly different. For the proposed method, it can be seen that the detection

performance is generally improving with the increasing of the attack size. For the strong attack group, the performance does not increase largely since it already maintains a very high-level accuracy when the attack size is small. For the moderate attack group, when the attack size is above 40%, its precision and recall values are both above 95%. Even for the weak attack group, its detection performances improve significantly. This is due to the fact that when the number of the fraudulent rates increases, more users share the similar behaviors so that they have a higher chance to be selected. For the BRS, however, the detection performance is generally decreasing with the increasing of the attack size, in particular for the strong attack group. It is since when the attack size is increasing, there are more fraudulent raters than the honest users in the system. The BRS, which relies on the majority rating in the systems, will be biased in its judgment. For the weak attack group, since the ratings from the fraudulent raters are always close to the honest users, its performance is less affected by the increasing number of the malicious users. The results confirm the vulnerability of the BRS in the Sybil Attack.

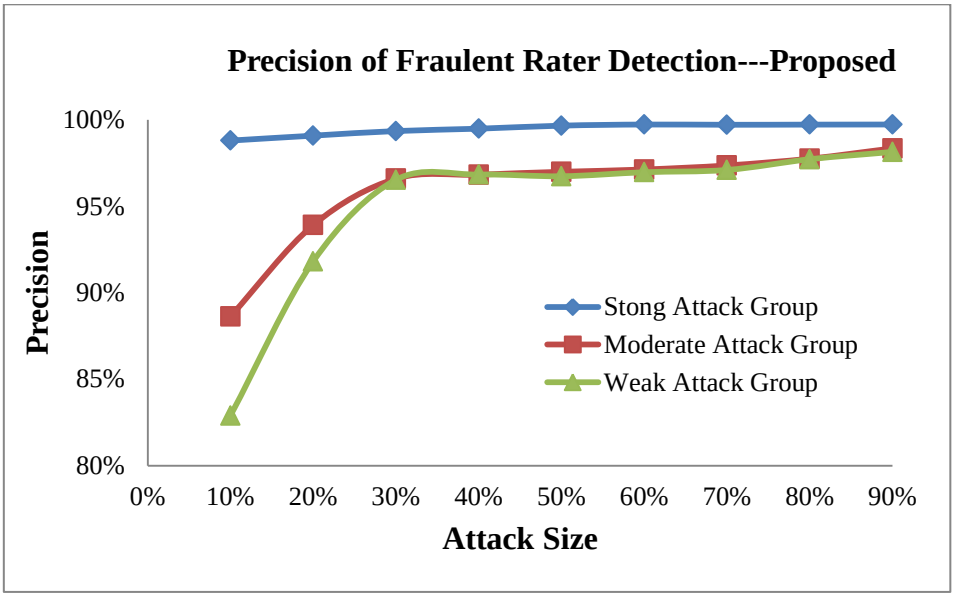

Figure 4.7(a) Precision of the Proposed Method in Various Attack Sizes

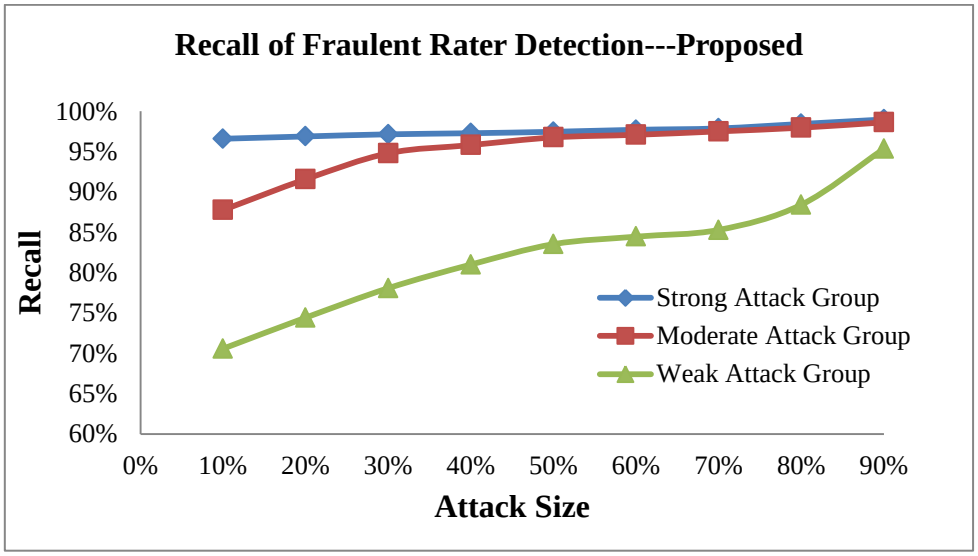

Figure 4.7(b) Recall of the Proposed Method in Various Attack Sizes

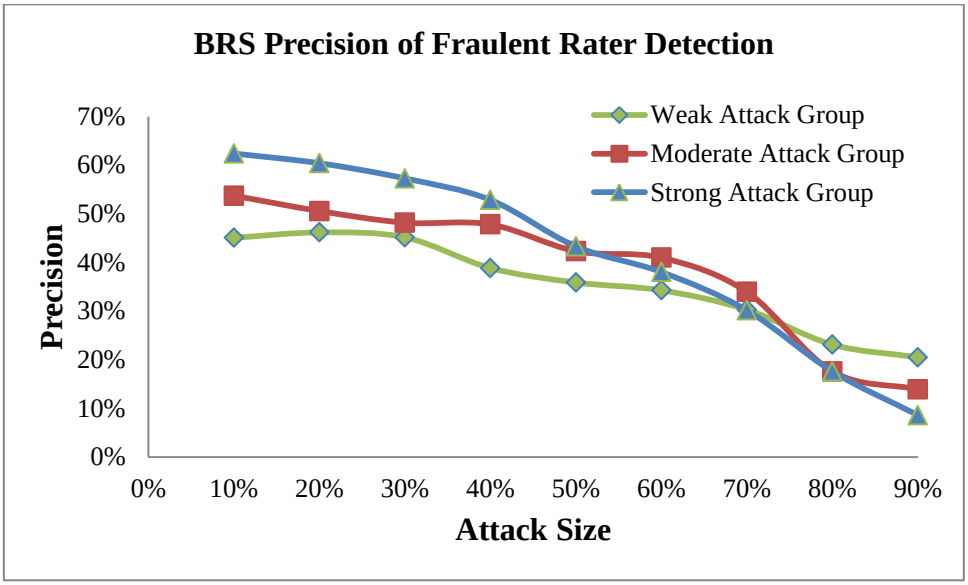

Figure 4.8(a) Precision of BRS in Various Attack Sizes

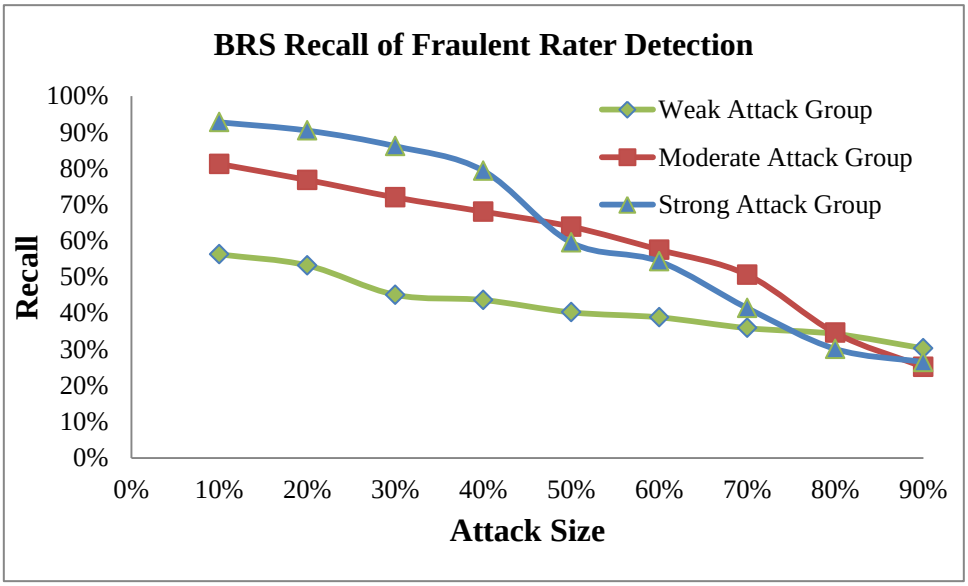

Figure 4.8(b) Recall of BRS in Various Attack Sizes

We then evaluate how the change of the intrusion size affects the performance by generating attacks files in various intrusion sizes. Intrusion size measures the ratio of all the entities who have been rated by the fraudulent raters. Seven different intrusion sizes are used,

of which vary from 30% to 90% in steps of 10%. Since the performance of intrusion size below 0.3 have already been largely covered in the single attack file testing, we do not discuss them in this subsection.

Similarly, we construct attack files for every intrusion size in all three attack level groups respectively. For example, suppose the intrusion size is 30%, which means that fraudulent raters should rate 90 entities. In each attack group, we randomly select a N attack files with the number of fraudulent raters which have rated a total of $(N\text{-}1+90)$ entities, denoted by AF_1, AF_2,.....AF_N. From AF_2 to AF_N, their ratings for the target entity e_1 are removed. For the non-target entities rated in these files, we will adjust their entity IDs if they are already existed in the AF_1. Then we change the user IDs in these files to make sure that they are consistent with those in the AF_1. Finally, the rating records in AF_2 to AF_N are added to the AF_1, and the detection performance is examined for this merged file. For each intrusion size, we repeat the experiments 500 times. Figure 4.9 and Figure 4.10 show the mean values of precision and recall for the proposed method respectively. We do not show the results for the BRS here since the intrusion size has limited impact on the BRS method. Therefore, the performance of the BRS in both precision and recall is relatively stable, as confirmed by the experiments. But we can see that compared to the attack size, intrusion size has a larger impact on the method performance, since a larger number of injected ratings make the fraudulent raters more distinguishable from the honest users. When the intrusion size is above 60%, the method can detect fraudulent raters accurately in every attack group. Additionally, the larger the intrusion size is, the wider the difference between the normal and

the malicious users. Thus, the intrusion size is positively correlated with the performance of the method in terms of both precision and recall.

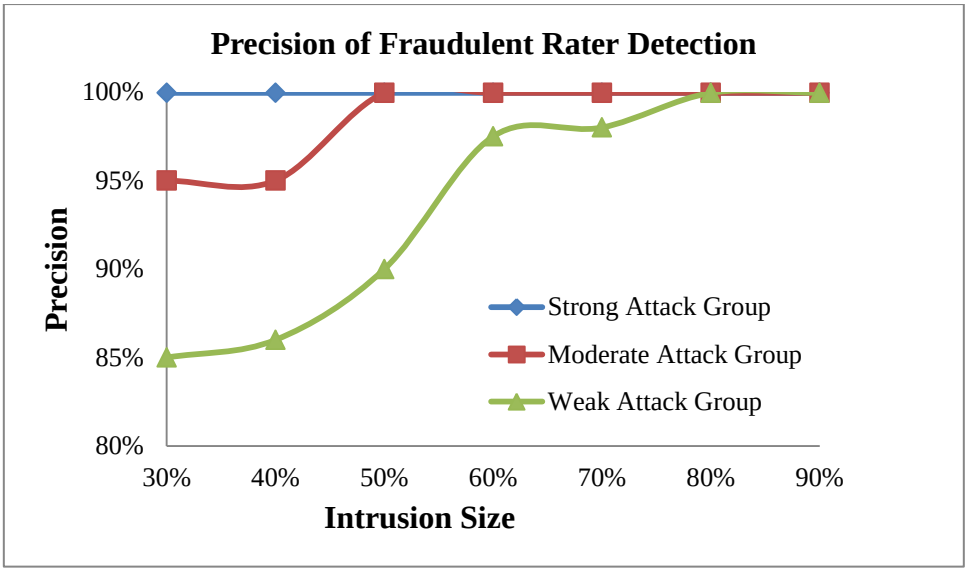

Figure 4.9 Precision of Fraudulent Rater Detection in Various Intrusion Sizes

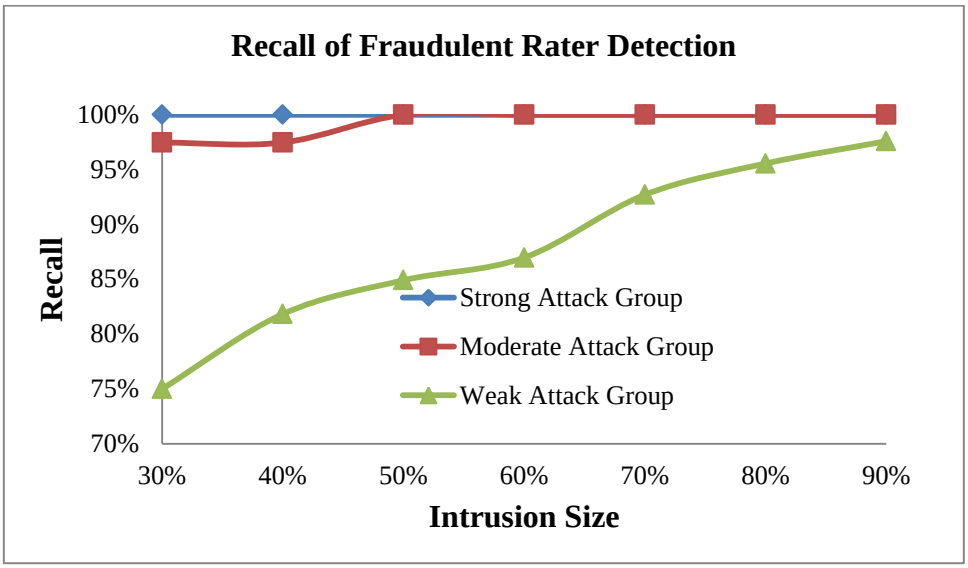

Figure 4.10 Recall of Fraudulent Rater Detection in Various Intrusion Sizes

4.5. Conclusion

Due to the anonymity in the Internet, it may be risky to interact with unfamiliar items or strange sellers. Rating systems have been shown to be effective for customers to judge the quality of the object and reduce the interaction-specific risk. However, rating systems are vulnerable to rating fraud, which will mislead the customers and further affect their motivation in participating into the future interaction. In this paper, we address the rating fraud issue and design the fraudulent rater detection method to improve the reliability of rating systems. By discovering certain temporal characteristics of both the target entity rating series and the fraudulent rater rating series, we could discriminate the target entity and cluster the corresponding dishonest raters. Several experiments are conducted to validate the performance of the proposed method by using real-world cyber competition data. While the benchmarking method suffers from the low precision, our algorithm could yield better overall detection performance. Moreover, our methods have shown its robustness in various attack environments including the Sybil Attack, Consistent Attack and Camouflage Attack. The method proposed in this study could facilitate the organizations relying on the rating systems for their better customer retention. It could also help reduce the financial risk associated with the e-commerce transactions.

We do not address the Whitewashing Attack in the present study due to data limitation. We believe, however, if the whitewashing fraudulent raters are using collaborative rating fraud strategy, our method can still discriminate them. In the future study, whitewashing fraudulent rater detection will be examined by simulation. In addition, our study focused on the collaborative rating fraud since it has more significant impact on the systems. Thus, our proposed method may be vulnerable to the singleton rating fraud, which

means the fraudulent rater does not have collaborators but just inject the unfair rating independently. For singleton fraudulent raters, they can be detected by examining the behavior deviation from the general distributions. The development of a more comprehensive detection framework by strategically combining our method with other research streams is also planned for future analysis.

CHAPTER 5. CONCLUSIONS AND FUTURE DIRECTIONS

Nowadays more and more organizations are applying intelligent systems to facilitate their decision making. Intelligent systems are widely adopted in various industry domains such as banking, retailing and entertainment. Like the human being, intelligent systems are capable to collect, process and analyze information in various problem domains. Due to the powerful computational capability, intelligent systems are advantageous in providing accurate, consistent and efficient decision making. However, the decision generated from intelligent systems highly relies on the input information. On account of diverse reasons, e.g. human errors or manipulations, the inputs of intelligent systems might be malicious so that the performance would face severe challenges. To maintain the smooth performance of intelligent systems, it is necessary to address the malicious input issues and design the corresponding defense strategies to accommodate malicious users.

This dissertation endeavors to discuss the malicious user detection schemes in three different types of intelligent systems: intelligent expert systems, intelligent recommender systems and intelligent rating systems. Each type of intelligent systems is designed for a unique purpose and is embedded with the distinct algorithm. Accordingly, the behavior pattern of the malicious user is different in each type of system and it results in three categories of malicious users: liars in the intelligent expert systems, shilling attackers in the intelligent recommender systems and fraudulent raters in the intelligent rating systems. Each of the three essays from Chapter 2 to Chapter 4 analyzes the influence of one category of malicious users on its systems' performance, explores its features of users' behavior patterns, develops the detection mechanisms based on the features, and evaluates the effectiveness of

the proposed methods with empirical experiments. For all proposed detection methods, the general objective is to discriminate the malicious users in the systems accurately. Usually, for organizations with intelligent systems, higher detection accuracy of malicious users indicates larger profit margins. However, that is not always the case. Hereby, the development of the detection methods considering the profit issue is also discussed when there is a tradeoff between detection accuracy and profit margin. The effectiveness of variant detection methods in this dissertation confirms the necessity of differentiation among problem structures. In this chapter, it concludes the dissertation with main findings in each essay and the discussion of their contributions to the malicious input research area and future research direction.

5.1 Conclusions

In this subsection, we summarize the main findings and conclude each of the three essays.

The first essay addresses the input distortion issue in intelligent expert systems. The methods proposed in this study explicitly differentiate liars from truth-tellers and treat them differently when their information is provided to redesign deductive expert systems. Two of the proposed methods, i.e., Split Tree (ST) and Consolidated Tree (CT), attempt to improve the accuracy of recommendations, and the other two, i.e., Value-based Split Tree (VST) and Value-based Consolidated Tree (VCT), aim to minimize the expected misclassification cost resulting from incorrect recommendations. Experimental results show that the proposed methods may lead to significant better accuracy or lower costs than that of the existing methods. In addition, the most accurate recommendation is not always the one with the lost

misclassification cost, since the recommendation of the accuracy based methods and that of the value-based methods are different given by the same input vector. Between the two pairs of proposed methods, this study finds that CT consistently outperforms ST in improving the accuracy of recommendations, and VCT always performs better than VST in reducing the expected misclassification cost. It also finds that the KM method proposed by Jiang et al. (2005), which essentially assumes that all users are potentially liars and treats them in the same manner, is not effective when there is a clear separation of liars and truth-tellers in the underlying population. This finding further confirms the necessity of differentiating liars from truth-tellers when addressing input distortion by users.

The second essay explores the shilling attack issues in intelligent recommender systems. This study proposes an integrated Value-based Neighbor Selection (VNS) method. The VNS method aims to select proper neighbors for collaborative filtering recommendation systems so that it can maximize the e-retailer's profit while protecting the system from shilling attacks. Different from the previous filtering-based shilling attacker detection techniques, this study utilizes the discounting-based strategies to estimate the attacker probability of each user. Each neighbor's rating is discounted before being aggregated to generate recommendations for other users. To validate the performance of the proposed VNS model, a number of experiments are conducted and are compared with several benchmarks from both the accuracy and profit perspectives. The experimental results find that the method proposed in this study could yield better overall accuracy and higher profit gains. This is particularly true when the filler size is small. In addition, this discounting-based strategy is more effective than filtering-based strategies in reducing the false positive rate. Hence, the

shilling attackers can be detected and their impacts are discounted so that the recommendations are much closer to users' actual tastes.

The third essay studies the rating fraud issue in intelligent rating systems. This study proposes a two-phase method for fraudulent rater detection. For the first step, the suspicious entities are filtered out by conducting the time series analysis on their rating series. And for the second, the fraudulent raters are discriminated by using clustering based methods. Several series of experiments are conducted to validate the performance of the proposed method by using a real-world cyber competition data. It finds that the proposed method could yield better overall detection performance. In particular, it can lead to better precision significantly. Therefore, the rating system is not only protected from fraudulent raters, but the quality of entity ratings is also preserved. In addition, the proposed method has shown its robustness in various attack models including Sybil Attack, Consistent Attack and Camouflage Attack. It also finds that the advantage of the proposed method is particularly significant when the attack size or the intrusion size is high. This study further demonstrates the effectiveness of applying the temporal features for the rating fraud detection.

5.2 Contribution and Future Research Directions

From the methodology perspective, this dissertation proposes various novel methods for dealing with malicious inputs to fill the research voids in three different intelligent systems. The first essay is the first study which differentiates liars from truth-tellers and considers misclassification costs when dealing with input noises for intelligent expert systems. The provision of both accuracy-based and value-based methods gives firms the

flexibility to select the appropriate method based on the underlying misclassification cost structure. Specifically, when the misclassification cost matrix is asymmetric, the VCT method is the most preferred. When the misclassification cost matrix is approximately symmetric or the misclassification costs are very difficult to estimate, the simpler CT method should be adopted. The method proposed in the second essay confirms the necessity of considering both of the e-retailer's profit and the recommendation accuracy in the development of intelligent recommender systems. The method proposed in the third essay breaks the constraints of the precious detection methods. This study can facilitate the design of the intelligent rating system in diverse attack environments.

From the managerial implication perspectives, the methods proposed in these studies can help firms to select the proper strategy to maintain their customer relationship and their revenue margins in various scenarios. First, to deal with the input distortion issues in the intelligent expert systems, the methods in Chapter 2 are applicable to the challenges in this area. Although the proposed methods require the verification of user-provided attribute values, the cost of such validation can be controlled by selecting the attributes that are relatively easy to verify. As a result, the expected benefit of adopting the proposed methods should exceed the expected cost under most real-world applications. Given the wide application of expert systems in various problem domains, the proposed methods can potentially lead to significant financial saving for organizations. Second, to deal with the shilling attackers in the intelligent recommender systems, the methods in Chapter 3 are applicable to the challenges in this domain. From the customer retention perspective, the attackers could be detected so that the recommendations are much closer to users' actual

tastes. The customers may trust the firms, and follow their recommendations in possible future purchases. From the e-retailers' perspective, the proposed method can bring significant financial revenue to them. Third, to deal with the fraudulent raters in the intelligent rating systems, the methods in Chapter 3 are applicable to the challenges in this field. After removing the ratings from the dishonest users and leaving the ratings from the honest users in the systems, users will have more confidence on the rating quality in the systems and refer to them before deciding which entity to interact with. Hence, it could greatly reduce the financial risks of online interactions, build up secure and reliable trust between firms and customers and maintain firms' long term profit.

Based on methods presented in this dissertation, there are multiple potential directions that could be explored in the future research. One direction is extending the current methods to address more complex problem scenarios. First, the methods in Chapter 2 consider only two groups of potential users, i.e., liars and truth-tellers. Such a two-group model may not be sufficient to capture the heterogeneity among potential users. For instance, some users never lie, some may lie occasionally, while others who lie frequently. How to extend the current model to incorporate multiple groups of users is an interesting direction for further research. Furthermore, the proposed methods are computationally intensive, hence they may become impractical when the number of attributes or the number of states for the attributes are large. In the future study, one could simplify the methods to reduce their complexity. For instance, when computing the CT Table, it is possible that the accuracy may not degrade much even if it considers only a small subset of the possible true vectors given an observed vector. The computation time is significantly reduced if a subset of vectors can be identified and used in

constructing the trees. Second, attackers in Chapter 3 are simulated based on the same attack model each time, which means that all attackers have a similar behavior. However, in the real-world shilling attacks, attackers may adopt a mixture of various attack models, which will increase the dissimilarity among attackers. The ways to extend the current method to accommodate the existence of different types of attackers should be examined in the further study. Third, Chapter 4 focuses on the collaborative rating fraud so it may be vulnerable to the singleton rating fraud. Although they can be detected by examining the behavior deviation from the general distributions, the development of a more comprehensive detection framework by strategically combining the current method with other research streams is also planned for the future work. Finally, all methods presented in this dissertation are designed based on the complete, though malicious, input information. In the real-world, however, it is not always the case. For instance, the applicant may strategically hide certain unfavorable information or the user may never submit their feedbacks to the systems. If the information is not missing randomly, the accuracy of the decision will be affected even if the malicious inputs are removed completely. Thus, designing a more adaptive platform to accommodate both the malicious and the incomplete input is an important area of interest for the future analysis.

BIBLIOGRAPHY

Akoglu, L., & Faloutsos, C. (2010). "Valuepick: Towards a Value-Oriented Dual-Goal Recommender System," in *ICDM Workshop: Optimization Based Techniques for Emerging Data Mining Problems*, Sydney, Australia.

Asch, S.E. (1961). "Effects of Group Pressure Upon the Modification and Distortion of Judgments," *Groups, Leadership, and Men*, pp 177–190.

Azari, R., et al. (2003). Current Security Management & Ethical Issues of Information Technology. Hershey, PA: Idea Group Publishing.

Ba, S., & Pavlou, P. (2002). Evidence of the effect of trust building technology in electronic markets: Price premiums and buyer behavior. *MIS Quarterly*, **26**(3), 243-268.

Boylu, F. Aytug, H. & Koehler, G, J. (2010). Induction over Strategic Agents. *Information Systems Research*, **21**(1), 170-189.

Breese, J.S., Heckerman, D., & Kadie, C. (1998). "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Microsoft Research Technical Report MSR-TR-98-12*, Redmond, CA.

Brodley, C. & Friedl, M. (1999). Identifying Mislabeled Training Data. *Journal of Artificial Intelligence Research*, **11**,131-167.

Bromley D. B. (2001). Relationships between personal and corporate reputation. *European Journal of Marketing*, **35**(3), 316-334.

Brutlag, J.D. (2000). "Aberrant Behavior Detection in Time Series for Network Monitoring," *Proceedings of the 14th USENIX Conference on System Administration*, Berkeley, CA, pp. 139–146.

Buller. D. B & Burgoon. J. K. (1996). Interpersonal Deception Theory. *Communication Theory*. **6**(3):203–242.

Burke, R., Mobasher, B., Zabicki, R., & Bhaumik, R. (2005). "Identifying Attack Models for Secure Recommendation," *In Beyond Personalization: A Workshop on the Next Generation of Recommender Systems,* San Diego, CA.

Chatterjee, K., de Alfaro, L., Pye, I. (2008). Robust Content-Driven Reputation. *AISec '08 Proceedings of the 1st ACM workshop on Workshop on AISec,* New York, NY, USA. 33-42.

Chen, L., Hsu, F., Chen, M., & Y., H. (2008). "Developing Recommender Systems with the Consideration of Product Profitability for Sellers," *Information Sciences* (178), pp. 1032-1048.

Chirita, P.A., Nejdl, W., & Zamfir, C. (2005). "Preventing Shilling Attacks in Online Recommender Systems". *In WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management. ACM Press, New York, NY,* pp. 67–74.

Das, A., Mathieu, C., & Ricketts, D. (2010). "Maximizing Profit Using Recommender Systems," *World Wide Web*, Raleigh, NC.

Dellarocas, C. (2000). Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. *Proceedings of the 2nd ACM Conference on Electronic commerce,* 150-157.

Douceur, J. (2002). The sybil attack. *IPTPS '01 Revised Papers from the First International Workshop on Peer-to-Peer Systems*. Springer-Verlag, London, UK. 251-260.

Duan, Y., Edwards, J. S., & Xu, M. X. (2005). Web-Based Expert Systems: Benefits and Challenges. *Information & Management*, **42**, 799-811.

Fei, L., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., & Ghosh, R. (2013). Exploiting Burstiness in Reviews for Review Spammer Detection. *Proceedings of The International AAAI Conference on Weblogs and Social Media (ICWSM-2013),* Boston, USA.

Fellegi, I. & Holt, D. (1976). A Systematic Approach to Automatic Edit and Imputation. *Journal of the American Statistical Association,* **71**, 17-35.

George, J. F., Biros. D. P., Adkins. M., Burgoon. J & Nunameker. J.F. (2004). "Testing Various Modes of Computer-Based Training for Deception Detection." *Intelligence and Security Informatics:* 411-417.

Goel, V. (2014). User Growth for Twitter Starts to Slow, and Stock Dips. Available at *http://www.nytimes.com/2014/02/06/technology/twitters-share-price-falls-after-it-reports-4th-quarter-loss.html?_r=1*

Herlocker, J., Konstan, J., Terveen, L., & Riedl, J. (2004). "Evaluating Collaborative Filtering Recommender Systems," *ACM Transactions on Information Systems* (22:1), pp. 5–53.

Hoffman, D. L. Novak, T. P. & Peralta, M. (1999). Building Consumer Trust Online. *Communications of the ACM,* **42**(4), 80-85.

Hoffman, K. Dage, D. & Nita-Rotaru, C. (2007), "A Survey of attacks on Reputation Systems". *Computer Science Technical Reports*. pp. 1677.

Hofmann, T. (1999). "Probalistic Latent Semantic Analysis," *Proceeding of 15th Conference on Uncertainty in Artificial Intelligence*. K. B. Laskey, H. Prade (eds), Morgan Kaufmann, San Francisco, pp. 289–296.

Holmes, D. I. (1994). Authorship attribution. *Computers and the Humanities*. **28**(2), 87-106.

Houser, D., & Wooders, J. (2006). Reputation in auctions: Theory, and evidence from eBay. *Journal of Economics and Management Strategy*, **15**(2), 353-369.

Hu, N., Liu, L., & Sambamurthy, V. (2010). Fraud detection in online consumer reviews. *Decision Support Systems*, **50**(3), 614-626.

Hu, N., Pavlou, P., & Zhang, J. (2006). Can online reviews reveal a product's true quality? Empirical findings and analytical modeling of online word-of-mouth communication. *Proceedings of the 7th ACM Conference on Electronic Commerce*, 324-330.

Huang, Z., Zeng, D., & Chen, H. (2007). "Analyzing Consumer-Product Graphs: Empirical Findings and Applications in Recommender Systems," *Management Science* (53:7), pp. 1146–1164.

Irissappane, A. A., Jiang S., & Zhang, J. (2012). Towards a Comprehensive Testbed to Evaluate the Robustness of Reputation Systems against Unfair Rating *Attacks. UMAP Workshops, volume 872 of CEUR Workshop Proceedings.*

Jiang, Y., Shang, J., Kemerer, C. F., & Liu, Y. (2011). "Optimizing E-tailer Profits and Customer Savings: Pricing Multistage Customized Online Bundles," *Marketing Science* (30:4), pp.737–752.

Jiang, Z. Mookerjee, V. S. & Sarkar, S. (2005). Lying On the Web: Implications for Expert Systems Redesign. *Information Systems Research,* **16**(2), 131-148.

Jin, X., Zhou, Y., & Mobasher, B. (2004). "Web Usage Mining based on Probabilistic Latent Semantic Analysis," *Proceeding of 10th ACM SIGKDD International Conference Knowledge Discovery Data Mining,* ACM, New York, pp. 197–205.

Jindal, N., & Liu, B. (2008). Opinion Spam and Analysis. *Proceedings of First ACM International Conference on Web Search and Data Mining (WSDM-2008),* Stanford, California, USA.

Josang, A., & Ismail, R. (2002). "The beta reputation system," *in Proceedings of the 15th Bled Electronic Commerce conference*.

Josang, A., Ismail, R. & Boyd, C. (2007). A survey of trust and reputation systems for online service provision. *Decision Support Systems,* **43**(2), 618-644.

Keogh, E., Lin, J., Lee, S.H., & Herle, H.V. (2006). "Finding the Most Unusual Time Series Subsequence: Algorithms and Applications," *Knowledge and Information Systems*(11:1), pp. 1–27.

Krishnakumar, K. (2003). Intelligent Systems For Aerospace Engineering - An Overview. *NASA Technical Report, Document ID: 20030105746*.

Lakhina, A., Crovella, M., & Diot, C. (2005). "Mining Anomalies Using Traffic Feature Distributions," *2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. ACM, New York, pp. 217–228.

Lam, S., & Riedl, J. (2004). "Shilling Recommender Systems for Fun and Profit," *13th International WWW Conference*. ACM, New York, pp. 309–402.

Lee, J., & Zhu, D. (2012). "Shilling Attack Detection—a New Approach for a Trustworthy Recommender System," *INFORMS Journal of Computing* (24:1), pp. 117–131.

Li, X., & Hitt, L. M. (2008). Self selection and information role of online product reviews, *Information Systems Research*, **19**(4), 456-474.

Liao, H. (2005). Expert System Methodologies and Applications-a Decade Review from 1995 to 2004. *Expert Systems with Applications*, **28**, 93-103.

Lim, E., Nguyen, V., Jindal, N., Liu, B., & Lauw, H. (2010). Detecting Product Review Spammers using Rating Behaviors. *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM-2010, full paper)*, Toronto, Canada.

Liu, D. R., & Shih, Y. Y. (2005). "Integrating AHP and Data Mining for Product Recommendation based on Customer Lifetime Value". *Information and Management* (42), pp. 340–387.

Liu, Y., Sun, Y. L., & Yu, T. (2011). Defending Multiple-User-Multiple-Target Attacks in Online Reputation Systems. *Privacy, security, risk and trust (passat), 2011 IEEE third international conference on and 2011 IEEE third international conference on social computing (socialcom)*, Boston, MA. 425-434.

Liu, S.,Yu, H., Miao, C., & Kot, A. C. (2013). A fuzzy logic based reputation model against unfair ratings. *AAMAS '13 Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*,Richland, SC. 821-828.

Mehta, B., Hofmann, T., Fankhauser, P. (2007). "Lies and Propaganda: Detecting Spam Users in Collaborative Filtering," *12th International Conference on Intelligent User Interfaces*. ACM, New York, pp. 14–21.

Metzger, M. J. (2004). Privacy, Trust, and Disclosure: Exploring Barriers to Electronic Commerce. *Journal of Computer-Mediated Communication,* 9(4).

Mobasher, B., Jin, X., & Zhou, Y. (2003). "Semantically Enhanced Collaborative Filtering on the Web," *1st Eur.Web Mining Forum.* B. Berendt, A. Hotho, D. Mladenic, M. van Someren, M. Spiliopoulou, G. Stumme (eds.), Lecture Notes in Computer Science (3209). Springer, Berlin, pp. 57–76.

Mobasher, B.,Burke, R., Bhaumik, R., & Williams, C. (2007). "Toward Trustworthy Recommender Systems: an Analysis of Attack Models and Algorithm Robustness," *ACM Transactions on Internet Technology* (7:4), pp. 1–40.

Mookerjee, V. S., & Dos Santos, B. L. (1993). Inductive Expert Systems Design: Maximizing System Value. *Information Systems Research,* **4**(2), 111-140

Mookerjee, V. S., Mannino, M. V. & Gilson, R. (1995). Improving the Performance Stability of Inductive Expert Systems Under Input Noise. *Information Systems Research.* **6**(4), 328-356.

Mukherjee, A., Kumar, A., Liu, B., Wang, J., Hsu, M., Castellanos, M., & Ghosh, R. Spotting Opinion Spammers using Behavioral Footprints. *To appear in Proceedings of SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2013)*, Chicago, USA.

O'Mahony, M., Hurley, N., Kushmerick, N., & Silvestre, G. (2004). "Collaborative Recommendation: a Robustness Analysis," *ACM Transactions on Internet Technology* (4:4) pp. 344–377.

Parker, S. (2011). 3 Tips for Spotting Fake Product Reviews – From Someone Who. Wrote Them. Available at *http://www.moneytalksnews.com/2011/07/25/3-tips-for-spotting-fake-product-reviews-%E2%80%93-from-someone-who-wrote-them/*

Power, D. J. (2000). Web-Based and Model-Driven Decision Support Systems: Concepts and Issues, *Proceedings' of the Americas Conference on Information Systems,* Long Beach, CA.

Quinlan, J. R. (1986). The Effect of Noise on Concept Learning. *Machine Learning, 2,* 149-166. R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (eds.) Morgan Kaufmann, Los Altos, CA.

Resnick, P., Kuwabara, K., Zeckhauser, R., & Friedman E. (2000). Reputation systems. *Communications of the ACM,* **43**(12), 45-48.

Resnick, P., & Zeckhauser, R. (2002). Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System. In M. R. Baye, editor, *The Economics of the Internet and E-Commerce, volume 11 of Advances in Applied Microeconomics.* Elsevier Science.

Rubin, D. (2004). Multiple Imputations for Nonresponse in Surveys. New York: Wiley.

Rudas, I. J., Fodor, J. (2008). Intelligent Systems: Plenary invited paper & workshop invited key lecture. *International Journal of Computers, Communications & Control,* 3, 132-138.

Sampson, B. (2007). "Sell Your Book on Amazon: Top-Secret Tips Guaranteed to Increase Sales for Print-on-Demand and Self-Publishing Writers," Outskirts Press, Parker, CO.

Schneider, J., et al. (2000). Disseminating Trust Information in Wearable Communities. *Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC2K),* Bristol, UK.

Shani, G., Brafman, R., & Heckerman, D. (2005). "An MDP-based Recommender System," *Journal of Machine Learning Research* (6), pp. 1265–1295.

Smith, C. (2014). 33 Amazing Amazon Statistics. Available at *http://expandedramblings.com/index.php/amazon-statistics/#.Uzs4A_ldWSo*

Soldo, F. (2011). Blacklisting Recommendation System: Using Spatio-Temporal Patterns to Predict Future Attacks. *IEEE Journal on Selected Areas in Communications,* **29**(7), 1423-1437.

Soldo, F., Le, A. & Markopoulou, A. (2010). Predictive Blacklisting as an Implicit Recommendation System. *INFOCOM, 2010 Proceedings IEEE,* San Diego, CA. 1-9.

Sorrel, C. (2009). Apple expels 1,000 apps after store scam. Available at *http://edition.cnn.com/2009/TECH/12/09/wired.apple.apps/index.html*

Sved, D. (2014). Nineteen companies found guilty of writing fake consumer reviews. Available at *http://www.heralddeparis.com/nineteen-companies-found-guilty-of-writing-fake-consumer-reviews/232920*

Teacy, W., Patel, J., Jennings, N. & Luck, M. (2006). Travos: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, **12**(2): 183-198.

Turban, E., & Aronson, J. E. (2000). Decision Support Systems and Intelligent Systems (6th ed). Prentice International Hall, Englewood Cliffs, NJ.

Wang, G., Xie, S., Liu, B., & Yu, P. S. (2011). Identify Online Store Review Spammers via Social Review Graph. *ACM Transactions on Intelligent Systems and Technology*, accepted for publication.

Whitby, A., Josang, A., & Indulska, J. (2004). Filtering out unfair ratings in bayesian reputation systems. *Proceedings of the 7th Int. Workshop on Trust in Agent Societies.*

Williams, C., Mobasher, B., & Burke. R, (2007). "Defending Recommender Systems: Detection of Profile Injection Attacks," *Service Oriented Computing and Applications* (1), pp, 157–170.

Williams, C., Mobasher, B., Burke, R., Sandvig, J., & Bhaumik, R. (2006). "Detection of Obfuscated Attacks in Collaborative Recommender Systems," *17th European Conference on Artificial Intelligence Workshop on Recommender Systems*. Riva del Garda, Italy.

Worsham, R. (2010). Penalties for a Falsified Credit Application. Available at http://www.ehow.com/Flist_6797816_penalties-falsified-credit-application.html.

Wu, X. & Zhu, X. (2008). Mining with Noise Knowledge: Error-Aware Data Mining. *Systems and Humans, 38,4,* 917-932.

Xie, S., Wang, G. Lin, S. Yu, P. S. (2012). Review Spam Detection via Temporal Pattern Discovery. *KDD '12 Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining,* New York, NY, USA. 823-831.

Yu, H., Kaminsky, M., Gibbons, P. B., & Flaxman, A. (2006). SybilGuard: defending against Sybil attacks via social networks. *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Press, New York, NY, USA. 267-278.

Zacharia, G., Moukas, A., & Maes, P. (2000). Collaborative reputation mechanisms for electronic marketplaces. *Decision Support Systems,* **29**(4), 371-388.

Zhang, S., Chakrabarti, A., Ford, J., & Makedon, F. (2006). "Attack Detection in Time Series for Recommender Systems. Proceeding of *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, pp. 809–814.

Zhang, Y. & Wu, X. (2010). Integrating Induction and Deduction for Noisy Data Mining. *Information Sciences, 180,* 2663-2673.

Zhou, L., Twitchell, D., Qin, T., Burgoon, J. & Nunamaker, J. (2003), "An Exploratory Study into Deception Detection in Text-Based Computer-Mediated Communication," *Proceedings of Thirty-Sixth Hawaii International Conference on System Sciences.*

Zhou, L., Burgoon, J., Twitchell, D., Qin, T. & Nunamaker, J. (2004). "A Comparison of Classification Methods for Predicting Deception in Computer-Mediated Communication," *Journal of Management Information Systems,* **20** (4), 139-165.

Zhou, L., Shi, Y., & Zhang, D. (2008), "A Statistical Language Modeling Approach to Online Deception Detection," *IEEE Transactions on Knowledge and Data Engineering,* **20** (8), 1077-1081.

Zhou, L. & Zhang. D. (2008). "Following Linguistic Footprints: Automatic Deception Detection in Online Communication". *Communications of the ACM,* September, 19-22.

Zhu, H., Xiong, H., Ge, Y., & Chen, E. (2013). Ranking fraud detection for mobile apps: a holistic view. *CIKM '13 Proceedings of the 22nd ACM international conference on Conference on information & knowledge management,* New York, NY, USA. 619-628.

Zhu, X. Wu, X. & Chen, Q. (2003). Eliminating class noise in large datasets. *Proc. International Conference on Machine Learning,* 920-927.

Zhu, X. Wu, X. & Yang, Y. (2004). Error detection and impact-sensitive instant ranking in noisy datasets. *Proc. Association for the Advancement of Artificial Intelligence,* 378-384.