A Process Pattern Mining Framework for the Detection of Health Care Fraud and Abuse

A Thesis

Submitted to the Faculty

Of

National Sun Yat-sen University

By

Wan-Shiou Yang

In Partial Fulfillment of the

Requirements for the Degree

Of

Doctor of Philosophy

June, 2003

TABLE OF CONTENTS

ABSTRACT

With the intensive need for health insurances, health care service providers' fraud and abuse have become a serious problem. The practices, such as billing services that were never rendered, performing medically unnecessary services, and misrepresenting non-covered treatments as medically necessary covered treatments, etc, not only contribute to the problem of rising health care expenditure but also affect the health of patients. We are therefore motivated to investigate the detection of service providers' fraudulent and abusive behavior.

In this research, we introduce the concept of clinical pathways and thereby propose a framework that facilitates automatic and systematic construction of adaptable and extensible detection systems. For the purposes of building such detection systems, we study the problems of mining frequent patterns from clinical instances, selecting features that have more discriminating power and revising detection model to have higher accuracy with less labeled instances.

The performance of the proposed approaches has been evaluated objectively by synthetic data set and real-world data set. Using the real-world data set gathered from the National Health Insurance (NHI) program in Taiwan, the experiments show that our detection model has fairly good prediction power. Comparing to traditional expense driven approach, more importantly, our detection model tends to capture different fraudulent scenarios.

LIST OF FIGURES

LIST OF TABLES

Chapter 1
Introduction

1.1 Motivation

Health care has become a major focus of concern and even a political, social, and economic issue in modern society. The medical resources required to meet public demand for high-quality and high-technology services, with consequent higher expenditures, are substantial, especially since the average length of life is increasing. People rely on government-sponsored and -managed health insurance systems, such as in Australia, France, and Taiwan, private health insurance systems, or both to share the expensive health care costs.

With such an intensive need for health insurances, fraudulent and abusive behavior, in the meantime, became a serious problem. For example, according to a report [WA96] to Congress by the General Accounting Office (GAO), health care fraud and abuse cost United States around 10% of its annual spending on health care. Since the annual national health care expenditure reached up to a trillion dollars in the US, the loss due to fraud and abuse is as high as $100 billions. Similar problem is reported in the health insurance programs of other developed countries [LLM97].

Health care fraud and abuse involve three parties, namely service providers, insurance subscribers, and insurance carriers. The practices such as billing services that were never rendered, performing medically unnecessary services, misrepresenting non-covered treatments as medically necessary covered treatments, and misrepresenting applications for obtaining lower premium rate, clearly contribute to

the immense problem of rising health care expenditure.

Of the three parties that participate in the health care fraud and abuse, service providers seem to cause the greatest damage, according to a report [NHCAA02] by National Health Care Anti-Fraud Association (NHCAA). Some types of fraud schemes (e.g., surgeries, invasive testing, and certain drug therapies) even place their trusting patients at significant physical risk and affect patients' health. For all the reasons mentioned above, we are motivated to investigate the detection of service providers' fraudulent and abusive behavior, and thereby propose this research.

1.2 Problem statement and the proposed approach

Detecting service providers' fraud and abuse needs intensive medical knowledge, and currently this task is often conducted by experts that manually review insurance claims and identify suspicious ones. Most computerized systems that are intended to help detect the undesired behavior still rely on experts' experiences in selecting statistically significant features so as to develop the core of detection models. As a result, the process of claim reviewing or system developing is a time- consuming effort, which is especially true in the case of large-scaled insurance systems, such as national insurance systems.

In this dissertation, we seek to develop an automatic approach so that the manual and ad hoc elements of the detection can be eliminated to a large extent. This approach must also be general such that the same set of developed tools can be readily applied to different data sources. We consider the health care fraud and abuse detection as a data analysis process. The central theme of our approach is to apply data mining

techniques to the gathered data to compute models that accurately capture the normal and fraudulent behavior (i.e., patterns) of clinical instances. We develop a prototype, for Mining Clinical Instances for Health Care Fraud and Abuse Detection, abbreviated as MCI HCFAD. Using MCI HCFAD, the inductively learned model replaces the manual detection task. This automatic approach eliminates the need to manually analyze and encode patterns, as well as the guesswork in selecting statistic measures. It is a general approach in that the same set of tools can be applied to other data sources that exhibit similar features.

1.3 Overview of the research

The rest of this dissertation is organized as follows. Chapter 2 examines in more detail the problems of health care fraud and abuse. We review the representative research efforts in this context, and briefly describe our research ideas and framework. Chapter 3 describes the algorithms for mining frequent patterns from clinical instances, with an emphasis on the clinical pathways structure analysis. Chapter 4 discusses how to analyze and select translated features automatically. Chapter 5 develops a hybrid detection model that works for cases with a small number of labeled instances. In Chapter 6, we summarize the contributions of this research and point out the limitations of the proposed approaches.

Chapter 2
The problem and the related Work

In this chapter, we first examine the problem of health care fraud and abuse so as to clarify our research and application scope. We then review representative practice and research efforts in this context. Subsequently, our research ideas, which are principally derived from the concept of clinical pathways, are described. Based on the research ideas, we propose a framework for detecting service providers' undesired behavior.

2.1 Health care fraud and abuse

As described in Chapter 1, there are three parties involved in the processing of health insurance. The service providers, which are comprised of the medical doctors, the hospitals, and even the ambulance companies, give health care services. The insurance subscribers, or the patients, receive health care from the service providers. The insurance carriers receive regular premiums from their thousands of subscribers, and make the commitment to pay health care cost on behalf of their subscribers.

With the existence of the three parties, working definitions about health care fraud and abuse were developed by National Health Care Anti-Fraud Association's (NHCAA) Board of Governors and published in their *1991 Guidelines to Health Care Fraud* [NHCAA91], which are quoted below.

> "*Health care fraud is an intentional deception or misrepresentation made by a person, or an entity, with the knowledge that the deception could result in some*

unauthorized benefit to him or some other entities."

"*Health care abuse is the provider practices that are inconsistent with sound fiscal, business, or medical practices, and result in an unnecessary cost, or in reimbursement of services that are not medically necessary or that fail to meet professionally recognized standards for health care.*"

From the above definitions, it is easy to see that the undesired behavior could be performed by any of the three parties. Common fraudulent and abusive behaviors pertaining to each party are listed below [NHCAA02].

(1) **The service providers' fraud and abuse**, including:

- Billing services that were never rendered.
- Performing more expensive services and procedures.
- Performing medically unnecessary services solely for the purpose of generating insurance payments.
- Misrepresenting non-covered treatments as medically necessary covered treatments for purposes of obtaining insurance payments.
- Falsification of patients' diagnosis and/or treatments histories.

(2) **The insurance subscribers' fraud and abuse**, including:

- Misrepresenting application for obtaining lower premium rate.
- Falsification of records of employment/eligibility.
- Falsification of medical claims.

(3) **The insurance carriers' fraud and abuse**, including:

- Falsification of reimbursements.
- Falsification of benefit/service statements.

According to NHCAA [NHCAA02], of the three kinds of fraud and abuse behaviors, the one committed by service providers take the greatest toll in the United States. The same conclusion is also supported by the investigations in other countries [HWGH97]. Even worse, the perpetrators of some types of fraud schemes (e.g., surgeries, invasive testing, and certain drug therapies) deliberately and callously place their trusting patients at significant physical risk. Based on these two reasons, we chose the detection of service providers' fraudulent and abusive behavior as our research target.

We further investigate current health insurance programs. As stated in [Glaser91], current health programs can be classified into three kinds in accordance with their payment methods:

(1) **Fee-for-service**. This is the traditional form of billing in both ambulatory and hospital inpatient services, in which care provider itemizes each service on a bill after the completion of care. Since more items demand more payments, the health insurance programs that adopt this payment method, have the most serious damage from service providers. Due to its simplicity, fee-for-service is still the most popular payment method adopted in today's private and national insurance programs, such as the Medicare in Australia and the National Health Insurance (NHI) in Taiwan.

(2) **Case payment**. In this billing form, a fixed amount is paid for

providing all necessary care to each type of diseases, classified by diagnosis. Though a fixed payment reduces the possibility of wasting service, it has been occasionally used by subscribers and carriers to pay for certain predictable care, such as normal deliveries in obstetrics. In practice, few insurance programs use this method to pay certain care services.

(3) **Global budget**. This is a new form of billing, in which service providers have prepared budgets of all operating costs expected during the next year. After the committee rate regulator, and the insurance carrier examine it and accept the estimates of clinical work load and costs, the carrier pays the annual total in installments. Global budget has been welcomed by insurance carriers as the most effective method of cost containment, since the financial payments of service providers cannot exceed the financial capacities of the system and thus have little abusive behavior. They soon, however, fear that their subscribers could be underserviced, and seek some mechanism to monitor care quality. Currently, some insurance programs (with some extent of experiments) adopt this method.

In cost containment systems– case payment and global budget, service providers receive payments (or a budget limit) before giving care services. Instead of fraud and abuse, underservicing behavior thereby becomes the major concern of such insurance carriers. Our approaches will be suitable in health insurance programs adopting fee-for-service payment method, since our goal is to detect service providers' fraud and abuse.

2.2 Current status

Currently, detecting service providers' fraud and abuse heavily rely on knowledge in medical domain. In practice, carriers in nearly every insurance program around the world employ experts, who are pre-eminent in their specialty, to detect suspicious claims in their programs. Therefore, experts review medical claims, and according to patients' conditions and expedience of care services, verify the necessity of each service. Clearly, the task performed by human experts is a time-consuming effort, which is especially true in the case of large-scaled insurance programs, such as NHI in Taiwan.

In some research work [Sokol98, SGWRJ01, HWGH97, Hall96], statistic features are identified by experts' consultants and used in the subsequent development of induction schemes. The research work described in [Sokol98, SGWRJ01] and funded by Health Care Financing Administration (HCFA) and the Office of the Inspector General (OIG), was to discriminate between normal and suspicious claims. For each care service, such as chiropractic services, lab/radiology procedures, and preventive medical services, a distinct set of features is identified. An inductive model is accordingly developed to detect suspicious claims of a particular care service.

The goal of the work described in [HWGH97, Hall96] by the Health Insurance Commission (HIC) of Australia, was to detect service providers who are practicing inappropriately, such as those who service their patients by performing more services than necessary, see their patients more often than warranted, or even bill non-rendered services. In this work, after discriminating features are available (typically 25-30

features identified by specialists), a fuzzy-logic, neural-network-based induction algorithm is used to produce the detection model. The detection model is then used to tag suspicious service providers.

While the inductive schemes of above researches reduce the workload of human experts to some extent, an enormous knowledge engineering task of identifying statistic features still remains. Restricted by the manual and ad hoc nature of the development process, the resultant prototypes have limited extensibility and adaptability.

2.3 Clinical pathways

Healthcare professionals, managers and administrators, always seek to provide timely, high quality health services. However, as stated by John Ovretviet [Guinane97], the many potential benefits often fail to be realized due to poor project planning and management.

> "*People and perfect processes make a quality health service – a poor quality service results from a badly designed and operated process, not from lazy or incompetent health care workers.*"

To meet the need of providing high quality health services, managed care plans are desirable. The concepts of *clinical pathways* (or *integrated care pathways*) were thus initiated in the early 1990, and can be defined as below [HAIPAP98, Ireson97].

> "*Clinical pathways are multidisciplinary care plans, in which diagnosis and*

therapeutic intervention are performed by physicians, nurses, and other staffs for a particular diagnosis or procedure."

For example, Figure 2.1 shows a pathway of cholecystectomy [Guinane97]. The pathway begins with the preadmission process, which mainly involves preadmission testing and anesthesia consult, goes though a number of assessments, surgery, and physicians' orders, and ends with a follow-up visit in the surgeon's office.

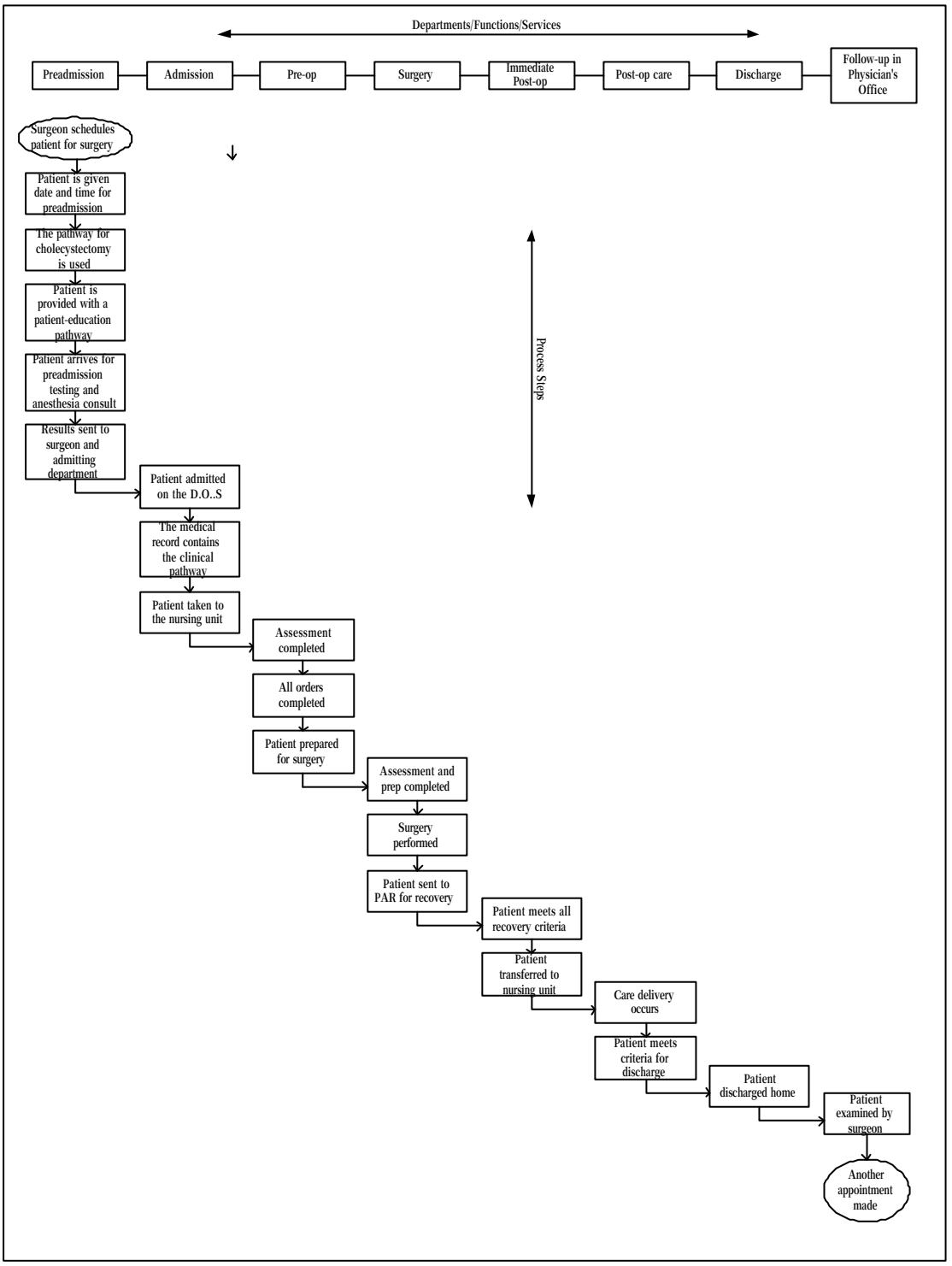


Figure 2.1 A pathway of cholecystectomy

Clinical pathways, as the one shown in Figure 2.1, are driven by physician orders, and clinical industry and local standards of care. Once the pathways are created, they are viewed as algorithms in that they offer a flow chart format of the decisions to be made

and the care to be provided for a given patient or patient group. Therefore, clinical pathways are developed for the following purposes [Guinane97]:

- Provide explicit and well-defined standards for care.
- Help reduce variations in patient care (standardize care).
- Help improve clinical outcomes.
- Support training.
- Provide a means of continuous quality improvement in healthcare.
- Support clinical audit.
- Support the use of guidelines in clinical practice.
- Help empower patients.
- Help manage clinical risk.
- Help improve communications between different care sectors.
- Disseminate accepted standards of care.
- Provide a baseline for future initiatives.
- Not prescriptive: don't override clinical judgment.

The application of clinical pathways is an efficient approach to analyzing and controlling clinical care processes. In today's competitive health care environment, due to the fact that competition advantage of a healthcare institution relies not only on outstanding professional quality but also on the agile clinical care processes, the concept of clinical pathways attracts much attention of managers in large hospitals around the world [Ireson97].

From the discussion above, it can be seen that clinical pathways aim to have medical staffs doing the care services in the *right order* [NELH]. Take the cholecystectomy

pathway in Figure 2.1 as an example. Care activities are sequenced on a timeline so that physicians can make suitable orders in accordance with the test results in the preadmission step; anesthetic can be executed during the performance of surgery on the basis of anesthesia consult. Best practice, without rework and resource waste, performs if the arrangement is in the right order.

This concept of clinical pathways shows great promise on detecting service providers' fraud and abuse. A care activity is very likely to be fraudulent if it orders suspiciously. For example, since physicians perform treatments in terms of test results, treatments following no diagnosis/test activities are doubtful; since physicians prefer performing simple, noninvasive tests before performing more complex and/or invasive tests, there is a high possibility that the same set of care activities performed in a different order is fraudulent or abusive.

Extensively, to accurately estimate the likelihood of a care activity performed on a particular patient, we must take into account the other activities performed on the patient. For example, while single ambulant visit is normal, repeating events are problematic, especially in the case that averaging length of pathway instances is small. On the contrary, a kidney transplant that rarely occurs should not be considered so unlikely if the patient has already undergone a series of diagnostic tests typically used to detect kidney disease.

Such observation, therefore, initiates an interesting idea that the clinical structures, including care activities and their execution orders, has the potential to discriminate

between normal and fraudulent practices. Explorative analysis of our dataset[1] supports this argument. In our data set, for example, about 40% fraudulent instances contain a structure of repeating ambulant visits while only 6% normal instances do so. In this research, we are thus motivated to exploit the discriminating power of clinical structures.

Confusion often arises over the differences between clinical pathways and packages of care. *Clinical pathways* are elements, or chunks of care service. Each chunk of service is developed into a clinical pathway, setting out detailed processes, i.e., a collection of activities, and done as a whole [NELH]. A *Package of Care* may contain one or more clinical pathways selected for a particular patient or target patient group. It describes the whole range of care given to that patient or patient group, usually for one episode of care.

Many factors, such as patients' conditions, physicians' preferences, and management cost may influence the selection of clinical pathways in a package of care given to a particular patient. Besides, different medical institutes often enforce different pathways, as there does not yet exist a universally best practice for a disease. Therefore, each patient may have a different practice (an instance). For our purpose–exploiting the clinical structures to discriminate between normal and fraudulent instances, it is necessary to find structures from practice instances since a complete definition of care package does not exist.

Therefore, we conceive to discover structures from practices–normal and fraudulent instances. To exploit the discriminating power of the discovered structures, we adopt

[1] Detailed descriptions of the data set are given in Chapter 4.4.

an induction scheme to construct the detection model. Based on the detection model, new coming instances can be detected automatically and systematically. In this research, we explore the advantages of knowledge discovery, rather than knowledge engineering, to detect service providers' fraud and abuse.

2.4 Research Framework

Motivated by the concept of clinical pathways, we propose a framework for detecting service providers' fraud and abuse. Generally, as shown in Figure 2.2, two sets of clinical instances, which are labeled as normal and fraudulent, serve as the input of structure pattern discovery module. The structure pattern discovery module produces a set of frequently occurred structure patterns, which then serve as features of clinical instances. Each clinical instance is seen as an example that comprises an assignment of features and a class label (normal or fraudulent). The resultant data set is further filtered by a feature selection module to eliminate redundant and irrelevant features. The selected features and the dataset are finally used to construct the detection model, performed by the induction module. The detection model will be used to detect the incoming instances that are fraudulent.

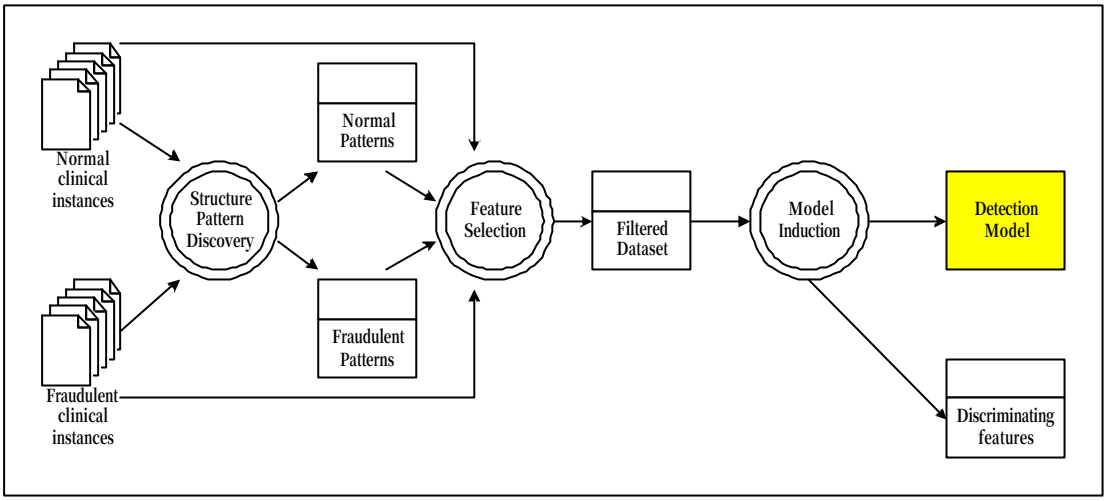


Figure 2.2 Research framework

In the research framework, we thereby identify three issues and form our series of investigations as below.

(1) **The problem of how to discover structure patterns from clinical instances**. As shown in Figure 2.1, a clinical pathway typically comprises a set of care activities. These activities, each appears over a temporally extended interval, may execute in a particular transition way, such as sequentially, concurrently, or repeatedly. A clinical instance, which consists of care activities from one or more clinical pathways, is thus formed as a set of activities in process. How to take these characters into account and design methods to efficiently discover structure patterns is the first problem we faced.

(2) **The problem of how to select relevant features**. Clearly, it is not the case that all discovered patterns have discriminating power. A certain percentage of patterns can be found in both normal and fraudulent cases and thus are irrelevant with respect to the detection problem. Also, a certain percentage of patterns are

correlated and thus form redundant features. How to efficiently eliminate these redundant and irrelevant features to improve the performance of the subsequent (induction) model construction is the second problem in which we are interested.

(3) **The problem of how to revise the detection model when the number of labeled examples is small**. The input to the proposed research framework consists of two sets of *labeled* instances, those classified by experts as normal and fraudulent cases. In practice, requiring a large number of labeled training examples to learn accurately is often prohibitive. Therefore, it is necessary to revise the existing detection model that is constructed from only labeled instances because they tend to learn less accurately for a small set of data. How to integrate other sources, such as unlabeled instances, to improve the accuracy of the detection model is another important issue in our research.

We investigate the issues listed above in order and report research results in the subsequent chapters.

Chapter 3
Structure pattern discovery

In order to construct the detection model described in antecedent chapter, we need to extract patterns in a way amenable to represent structures of clinical instances. In this chapter, we explore the entrance problem: the structure pattern discovery.

Typically, a clinical instance, as described in Section 2.3, is a process instance comprising a set of activities, each a logical unit of work performed by a medical staff. For example, a patient treatment flow may involve measuring blood pressure, examining respiration, and medicine treatment, just to name a few. These activities, each appearing over a temporally extended interval, may execute sequentially, concurrently, or repeatedly. For example, before giving any therapeutic intervention, diagnosis activities are often executed to verify conditions of a patient. Also, in order to gain better curative effect in some cases, it is necessary to execute a number of therapeutic interventions concurrently.

As a result, if we want to extract structure patterns from clinical instances, we need to take structural characteristics of process−temporally extended intervals and various transition ways−into consideration. We accordingly use a temporal graph to represent a clinical instance in our research. Detailed definitions of temporal graph and corresponding algorithms for discovering patterns are described in this chapter. The experimental results in evaluating performance of the proposed algorithms are also reported.

3.1 Related works

The works reported in [AGL98, Datta98, HY02] deal with the problem of discovering a process model from a set of process instances and assume the existence of a process model (i.e., control dependencies between activities) underlying a given set of process instances. In this vein, such discovery, using a directed graph [AGL98, HY02] or a finite state machine [Datta98] for representing process instances, aims at discovering a process model that best describes the set of process instances. Our study significantly differs from the process model discovery in that we do not assume the existence of an underlying process model but is designed to identify frequently observed temporal dependencies within process instances rather than control dependencies that are presumably genuine in the process instances.

Our work is closest to sequential pattern discovery that discovers frequent sequential occurrence of activities (e.g., items purchased) across transactions of the same entity (e.g., customer) [AS95, SA96]. The sequential pattern discovery is to find the maximal sequences among all sequences that have a certain user-specified minimum support. The work on sequential pattern discovery assumes that a transaction contains a set of activities occurring at the same time and that transactions of the same entity are sequentially ordered. While we assume that an activity appears over a temporally extended interval, two activities may overlap or occur in sequence, making sequential pattern discovery inappropriate because grouping activities into transactions cannot capture all possible temporal relationships between activities.

In addition, graph-based mining techniques are proposed in [CH00] that identifies interesting and repetitive substructures within structural data. Representing structural

data as a labeled graph, the substructure discovery techniques aim at finding all possible substructures from the graph. By its nature, the techniques discover only substructures that are regionally connected subgraphs and disregards transitive relationships among objects, limiting its applicability to our structure pattern discovery problem, where transitivity in temporally sequential relationships prevails.

Finally, [BWJ98] deals with the discovery of frequent-event patterns in a time sequence that consists of a set of time-stamped events. The discovery process starts with a user-specified event structure that consists of a set of variables representing events and temporal constraints between variables. Its goal is to identify instantiations of variables in the event structure that appear frequently in the time sequence. The event pattern discovery differs from our work in several ways. First, it assumes an event appears at a time point rather than over a time interval. Second, it searches for instantiations of a user-specified event structure rather than discovering all possible frequent temporal relationships among events within a time sequence.

3.2 Formalization of structure pattern discovery problem

A clinical instance comprises a set of activities, each of which is an execution unit that leads to the transition of state in the instance. The execution of an activity spans a temporally extended period. Each activity may also be associated with such information as execution entity(s) involved, execution location and execution outcome. However, since the main intent of this research is to discover frequent activities and their associated temporal dependencies, we make use only of the starting time and ending time of an activity execution. Our view on a clinical instance can be formally described as below.

Definition 3.1 A clinical instance I is a set of triplets (V_i, st, et), where V_i uniquely identifies an activity, and st and et are timestamps representing the starting time and ending time of the execution of V_i in I, respectively.

Given a clinical instance, the temporal relationship between any activity pair can be classified into two types: *followed* and *overlapped*.

Definition 3.2 In a clinical instance I, an activity V_i is *followed* by another activity V_j if V_j starts after V_i terminates in I.

Definition 3.3 In a clinical instance I, two activities, V_i and V_j, are *overlapped* if V_i and V_j incur overlapped execution durations in I.

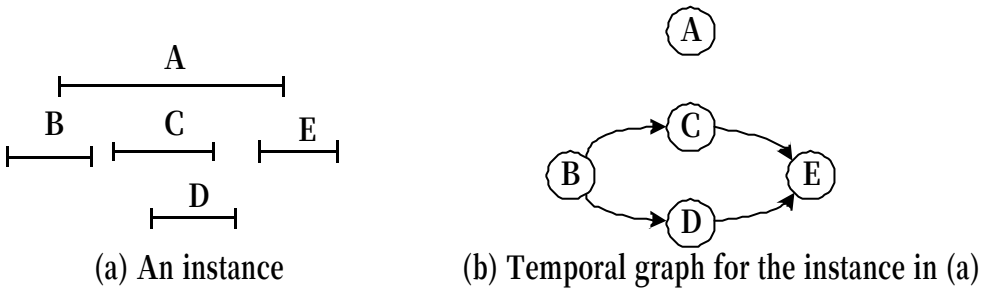
Definition 3.4 An activity V_i is *directly followed* by another activity V_j in a clinical instance I if V_i is followed by V_j in I and there does not exist a distinct activity V_k in I such that V_i is followed by V_k and V_k is followed by V_j in I.

To represent temporal relationships between activities in a clinical instance concisely, a *temporal graph* is defined as follows.

Definition 3.5 The pertinent *temporal graph* of a clinical instance I is a directed acyclic graph $G = (V, E)$, where V is the set of activities in I, and E is a set of edges. Each edge in G is an ordered pair (V_i, V_j), where V_i, $V_j \in V$, $V_i \neq V_j$, and V_i is directly followed by V_j.

Transforming a clinical instance into its corresponding temporal graph representation is straightforward. We first traverse the activities in the given clinical instance by the

ascending order of their starting times. For each activity *a*, the set *F* of activities that directly follow *a* are identified. Subsequently, edges connecting *a* to each activity in *F* are created. As shown in Figure 3.1(a), activity *B* will be processed first due to its earliest starting time among all activities in the instance. Activities *C* and *D* directly follow *B*; thus, two edges are created from *B* to *C* and *D*, respectively, as shown in Figure 3.1(b). The subsequent traversal of this clinical instance processes activities *A*, *C*, *D*, and *E* in sequence. The resulting temporal graph corresponding to this instance is graphically illustrated in Figure 3.1(b). From a given temporal graph *G*, it is evident that an activity V_i is followed by another activity V_j if there exists a path from V_i to V_j in *G*, and V_i and V_j are overlapped otherwise. As shown in Figure 3.1(b), activity *B* is followed by *E* since there exists a path from *B* to *E*. In contrast, activities *A* and *B* are overlapped since there does not exist a path that connects them.



(a) An instance (b) Temporal graph for the instance in (a)

Figure 3.1 Example of a clinical instance and the corresponding temporal graph

A structure pattern can also be represented as a temporal graph that has a certain user-specified minimum support.

Definition 3.6 A temporal graph *G* is said to be *supported* by a clinical instance *I* if all followed and overlapped relationships that exist in *G* are present in *I*.

Definition 3.7 A temporal graph *G* is said to be *frequent* if it is supported by no less than *s*% of the clinical instances, where *s*% is a user-defined minimum support

threshold.

Definition 3.8 A temporal graph $G=(V, E)$ is a *temporal subgraph* of another temporal graph $G'=(V', E')$ if $V \subseteq V'$ and for any pair of vertices $v_1, v_2 \in V$, there is a path in G connecting v_1 to v_2 if and only if there is a path in G' connecting v_1 to v_2. If G is a temporal subgraph of G', then G' is a *temporal supergraph* of G.

Problem statement: Given a set of temporal graphs, each of which represents a clinical instance, the structure pattern discovery is to find all frequent temporal graphs. Each such temporal graph is referred to as a structure pattern.

3.3 Structure pattern discovery algorithms

In this section, three different algorithms, namely TP-Graph, TP-Itemset, and TP-Sequence, are proposed for the described structure pattern discovery problem. TP-Graph directs its discovery process directly based on the temporal graph representation. On the other hand, TP-Itemset extends the Apriori algorithm that finds frequent itemsets [AS94] discovers patterns from a set of clinical instances, each of which is represented as a set of temporal relationships. Finally, in the TP-Sequence algorithm, each clinical instance is represented as a quasi-sequence where the overlapping and followed-by relationships of clinical instances are properly preserved. Accordingly, a sequential pattern discovery technique, specifically the AprioriAll algorithm [AS95], is extended to discover patterns from the set of quasi-sequences.

3.3.1 TP-Graph algorithm

As with association rule [AS94] and sequential pattern [AS95] algorithms, the TP-Graph algorithm exploits the downward closure property of the support measure

to improve the efficiency of searching for frequent temporal graphs. The downward closure property suggests that if a temporal graph G has support of at least s%, any temporal subgraph of G must have a support of at least s% or, conversely, if a temporal graph G has a support of less than s%, any temporal supergraph of G definitely will have support of less than s%. Accordingly, we adopted an iterative procedure similar to that in the Apriori [AS94] and AprioriAll [AS95] algorithms. Specifically, potentially frequent temporal graphs (or called *candidate temporal graph*) of size k can be constructed from joining frequent temporal graphs of size $k-1$. The clinical instances are then scanned to identify frequent temporal graphs of size k from the set of candidate temporal graphs of the same size. This procedure is iteratively executed until no further frequent temporal graphs can be found. Let C_k and L_k denote the set of candidate temporal graphs and the set of frequent temporal graphs of size k, respectively. Each iteration k performs the following two steps whose challenges and solutions are detailed in the following subsections, respectively.

1. If k=1, C_k is the set of all single-activity temporal graphs. Otherwise, join in pair-wise the frequent temporal graphs of size $k-1$.
2. Scan the clinical instances to determine L_k from C_k.

Joining frequent temporal graphs

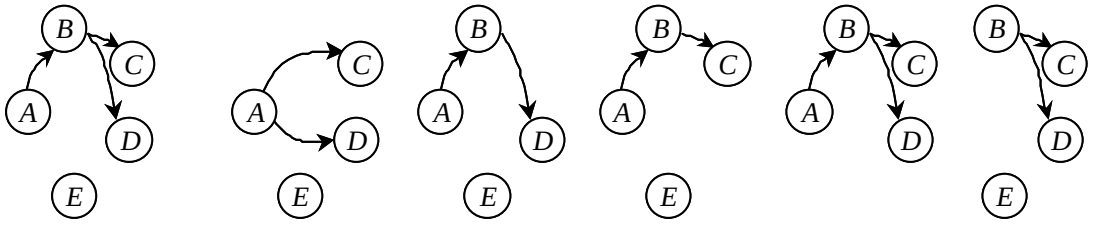
Intuitively, two frequent temporal graphs of size $k-1$ can be joined if they differ only in one activity and contain the same temporal relationships for any pair of common activities. However, this simple-minded joining process will result in many redundant candidate temporal graphs. Consider the following example. Suppose the set of frequent temporal graphs in iteration 2 be $\{A \rightarrow B$[2], $B \rightarrow C$, $A \rightarrow C\}$. Any pair in the set

[2] This simplified representation differs from the temporal graph representation defined in Definition 3.5. Here, $A \rightarrow B$ denotes a temporal graph consisting of activities A and B where A is directed followed

can be joined to form the candidate temporal graph of $A \rightarrow B \rightarrow C$. That is, three identical candidate temporal graphs of size 3 will be generated. In the following, a joining algorithm is proposed to eliminate or control such redundancy.

Definition 3.9 Let G be a temporal graph and v be a vertex in G. The operation of substracting v from G, denoted as $G-\{v\}$, deletes v and its associated edges from G. In addition, transitive edges via v are reconstructed by connecting each source vertex of incoming edges of v to each destination vertex of outgoing edges of v.

This subtraction operation can be illustrated as follows. Figure 3.2(b)-(f) show all of the temporal subgraphs resulted from subtracting a vertex from the temporal graph G shown in Figure 3.2(a). When the vertex B is subtracted from G, edges $A \rightarrow C$ and $A \rightarrow D$ are reconstructed as shown in Figure 3.2(b). As shown in Figure 3.2(c), the deletion of the vertex C from Figure 3.2(a) does not introduce any new edge in G since C does not have any outgoing edge. Figure 3.2(d), (e), and (f) illustrate the remaining temporal subgraphs derived from Figure 3.2(a) by deleting D, E, and A, respectively.



(a) Temporal Graph G (b) $G-\{B\}$ (c) $G-\{C\}$ (d) $G-\{D\}$ (e) $G-\{E\}$ (f) $G-\{A\}$

Figure 3.2 Examples of subtraction operation

Observation 3.1 Let s be a vertex without incoming edges (called a source vertex)

---

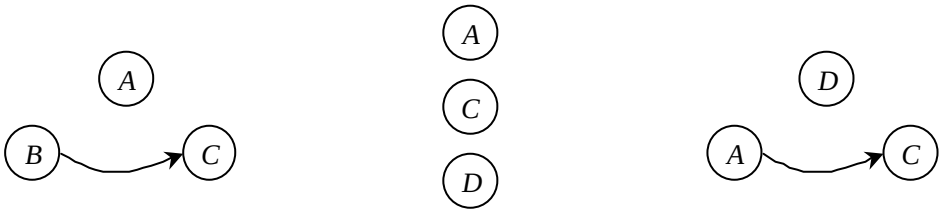
by B.

and e be another vertex without outgoing edges (called a sink vertex) in a temporal graph G. If G is frequent, both $G-\{s\}$ and $G-\{e\}$ must be frequent.

Based on this observation, to determine whether two frequent temporal graphs can be joined, only their source vertices and sink vertices need to be considered. Accordingly, we formally define joinable temporal graphs as follows.

Definition 3.10 Two temporal graphs G_i and G_j are said to be *joinable*, if there exists a source vertex s in G_i and a sink vertex e in G_j such that $G_i-\{s\} = G_j-\{e\}$.

Consider the temporal graphs shown in Figure 3.3. Designating vertex B as a source activity of G_1 shown in Figure 3.3(a) and vertex D as a sink activity of G_2 shown in Figure 3.3(b), these two temporal graphs are *joinable* since $G_1-\{B\} = G_2-\{D\}$. The temporal graphs G_1 and G_3 or G_2 and G_3, however, are not joinable.



(a) Temporal Graph G_1 (b) Temporal Graph G_2 (c) Temporal Graph G_3

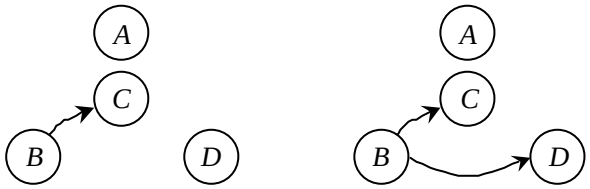
Figure 3.3. Three example temporal graphs of size 3

Given two joinable temporal graphs G_i (with s being a source vertex) and G_j (with e being a sink vertex), the temporal relationship between any pair of activities (except that between s and e) present in G_i or G_j will be preserved in a resulting candidate temporal graph. Since more than one permissible temporal relationship between s and e may exist, the joining of two joinable temporal graphs of size k–1 can lead to

multiple candidate temporal graphs of size k. The temporal relationship between s and e in a candidate temporal graph can be: 1) no edge exists between s and e or 2) an edge connects s to e. Note that the case where an edge connects e to s needs not be considered, as it results in a temporal graph with s and e not being source and sink vertices respectively. From Observation 3.1, it is clear that if a temporal graph G with a source vertex s and a sink vertex e is frequent, both frequent temporal graphs $G-\{s\}$ (where s is a source vertex in G) and $G-\{e\}$ (where e is a sink vertex in G) must be joinable. Formally, the join set of two joinable temporal graphs G_i (with s being the source vertex) and G_j (with e being the sink vertex) is composed of

1. $G_i \cup G_j$[3], and
2. $G_i \cup G_j \cup \{s \rightarrow e\}$ if there does not exist a path from s to e in $G_i \cup G_j$.

Consider the two joinable temporal graphs G_1 and G_2 shown in Figure 3.3. The join set of G_1 (with B being a source vertex) and G_2 (with D being a sink vertex) includes two candidate temporal graphs of size 4 as shown in Figure 3.4.



(a) (b)

Figure 3.4 Two Candidate Temporal Graphs Resulting from Joining G_1 and G_2 in Figure 3.3

The described downward closure property can further be exploited to reduce the set of resulting candidate temporal graphs. A candidate temporal graph G of size k will not be frequent if any of its temporal subgraphs of size $k-1$ is not in L_{k-1} and, hence,

[3] The union of two graphs $G_i=(V_i, E_i)$ and $G_j=(V_j, E_j)$ results in a new graph $G=(V_i \cup V_j, E_i \cup E_j)$.

should be eliminated from C_k. Such pruning process requires, for each candidate temporal graph of size k, the derivation (using the subtraction operation defined in Definition 3.9) of all of its temporal subgraphs of size $k-1$. The pseudo code of *GenerateCandidateGraph*() for generating a set of candidate temporal graphs of size k from a set of frequent temporal graphs of size $k-1$ and that of *DeriveSubgraph*() for deriving all temporal subgraphs of size $|G|-1$ for a temporal graph G are listed below.

```
GenerateCandidateGraph(a set of frequent temporal graphs: TGS): a set of temporal
graphs
{
    CandidateSet = Ø;
    For (each pair of graphs (Gi, Gj) in TGS) {
        For (each source vertex s in Gi) {
            For (each sink vertex e in Gj) {
                If (Gi−{s}= Gj−{e}) { /* joinable */
                    UG1 = Gi ∪ Gj; UG2 = Gi ∪ Gj ∪ {s→ e};
                    CandidateSet = CandidateSet ∪ {UG1};
                    If there exists no path from s to e in UG1
                    Then CandidateSet = CandidateSet ∪ {UG2};
                } /* end-if */
            } /* end-for*/
        } /* end-for */
    } /* end-for*/
    For (each graph G in CandidateSet) {
        If DeriveSubgraph(G) ∩ TGS ≠ DeriveSubgraph(G)
        Then CandidateSet = CandidateSet – {G};
    } /* end-for */
    Return CandidateSet;
}

DeriveSubgraph(a temporal graph: G): a set of temporal graphs
{
    Subgraph = Ø;
    For (each vertex v in G) {
        Source = the set of vertices incident to v;
        Sink = the set of vertices incident from v;
        SG = G – {v};
        For (each vertex pair (vs, vd) where vs ∈ Source and vd ∈ Sink) {
            If there does not exist a path between vs and vd in SG then SG = SG ∪{vs→
            vd};
        } /* end-for */
        Subgraph = Subgraph ∪ {SG};
```

```
    }
    Return Subgraph;
}
```

Scanning clinical instances

To find frequent temporal graphs from a set of candidate temporal graphs, we have to compute their support by scanning the set of clinical instances. To efficiently decide the set of candidate temporal graphs that a given clinical instance supports, we adopted the *hash-tree* data structure proposed by Agrawal and Srikant [AS94, AS95]. Use of the hash-tree to store candidate temporal graphs of the same size requires a total order on the vertices in each temporal graph. In this study, the vertices of each temporal graph are sorted based on its graph topology. A topological sort of graph G is a linear ordering of all its vertices such that if G contains an edge (u, v), then u appears before v in the ordering [CLR89]. To ensure a unique topological sort for a given temporal graph, the lexicographic order is applied to vertices that are temporally overlapped. The resulting order is called the temporal sequence of a temporal graph. For instance, the temporal sequence of the temporal graph shown in Figure 3.4(a) is $<B, A, C, D>$.

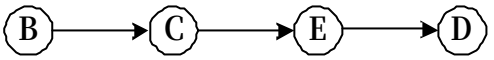
A node in the hash-tree either contains a set of temporal graphs (a leaf node) or a hash table (an interior node). Each bucket in the hash table of an interior node points to a child node. To insert a candidate temporal graph G, we start from the root and follow appropriate pointers until a leaf node is reached. At an interior node at depth d (assuming the depth of the root node of the hash-tree be 1 and that of a child node of an interior node at depth d be d+1), we decide which branch to follow by applying a hash function to the d-th vertex in the temporal sequence of G. Initially, the root node is a leaf node. When a leaf node L at depth d overflows (i.e., the number of temporal

graphs in the leaf node exceeds a specified threshold), L is converted to an interior node and several leaf nodes are created as the child nodes of L. All the temporal graphs originally stored in L are distributed to these leaf nodes by applying the hashing function to the d-th vertices in their temporal sequences.

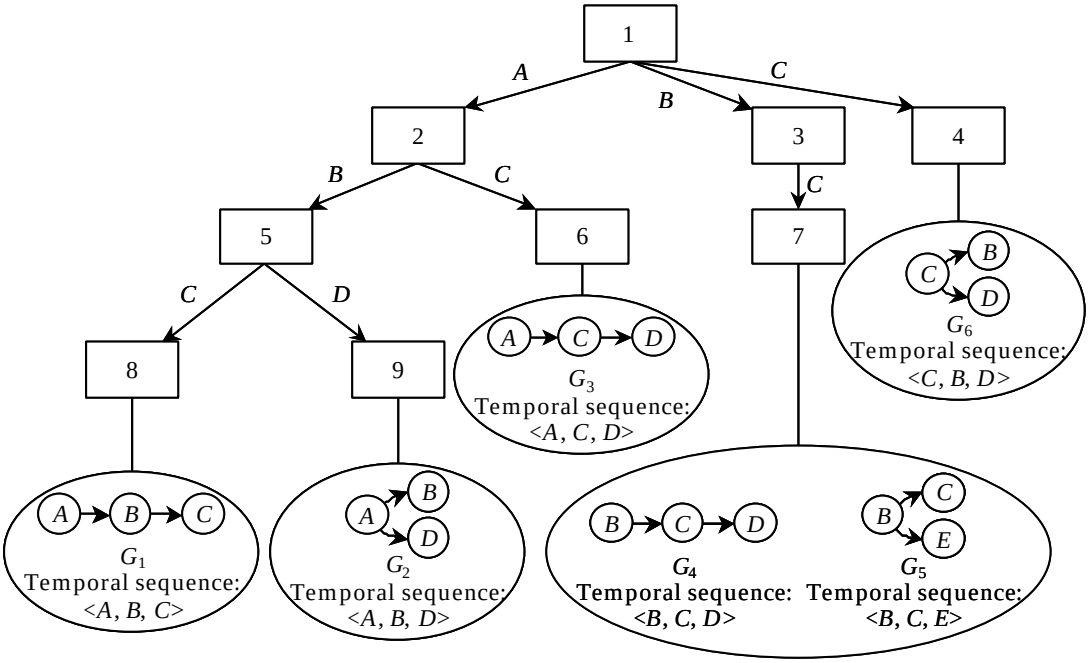
Such a hash-tree, once constructed, can be used to determine the subset of candidate temporal graphs that is supported by a given clinical instance I by traversing the hash-tree. Let S denote the temporal sequence of I. The traversal starts at the root node by applying the hashing function on every vertex in S to determine the set of nodes at depth 2 to visit. At an interior node to which a vertex a in S has just hashed, the hashing function is then applied to each vertex after a in S. The traversal process continues until leaf nodes are reached. At each leaf node reached, we determine which of the candidate temporal graphs in the leaf are supported by I and increment their support count by one. After scanning all the clinical instances, the candidate temporal graphs whose support exceeds the user specified minimum threshold form the set of frequent temporal graphs of this iteration.

Consider a segment of hash-tree for candidate temporal graphs of size 3 as shown in Figure 3.5. By hashing on every vertex in the temporal sequence <*B*, *C*, *E*, *D*> of the clinical instance I shown in Figure 3.5(a), we examine those nodes that start with *B*, *C*, *E*, or *D*, respectively. In this case, the nodes 3 and 4 will be visited next. At node 3 in the hash-tree shown in Figure 3.5(b), we can only hash on vertex *C*, *E* and *D* since we have reached node 3 by previously hashing on vertex *B*. As a result, node 7 will then be visited. On the other hand, since node 4 is a leaf node, whether its candidate temporal graph (i.e., G_6) is supported by I is then examined. Similarly, the temporal graphs (i.e., G_4 and G_5) in node 7 will be examined against I and the support of G_4 is

incremented by one since it is supported by *I*.



(a) A temporal graph for an instance



(b) A Segment of Hash-tree

Figure 3.5 Hash-tree for candidate temporal graphs of size 3

The pseudo code for inserting a candidate temporal graph into a hash-tree, named *AddOneGraph*(), and that for traversing the hash-tree for a given clinical instance, named *Traverse*(), are listed below. Note that *Traverse*() is a recursive function that takes three parameters, namely the current node *C* of the hash-tree, the target clinical instance *I*, and the position of the vertex in *I* that previously hashed to *C*. Initially, we call *Traverse*(*root* of the hash-tree, *I*, 0).

```
AddOneGraph(a hash-tree: T, a temporal graph: G with temporal sequence <v1, v2, ..., vn>)
{
    C = root of T; Level = 1; /* initialize */
    While C is not a leaf node of T do {
        C= Hash(v_Level );
        Level++;
    }
    Insert G into C;
    If C is full
    {
        Create a hash table H with each entry pointing to a new leaf node;
        For each temporal graph R in C
        {
            Assign R to a leaf node by hashing on its vertex at depth Level;
        } /* end-for */
        Assign the hash table H to C;
    } /* end-if */
}

Traverse(a node pointer: C, a clinical instance: I with temporal sequence <v1, v2, ..., vn>, previous position: s)
{
    If (C is a leaf node) {
        For (each temporal graph G in C) {
            If G is supported by I then G.count++;
        } /* end-for */
    }
    else {
        position = s;
        Do {
            NewC = Hash(v_position );
            Traverse(NewC, I, position+1);
            position++;
        } While (position ≤ n);
    } /* end-if */
}
```

Theorem 3.1 shows that this traversal procedure indeed returns the desired result.

Lemma 3.1 For each candidate temporal graph G supported by a clinical instance I, the temporal sequence of G must be a subsequence[4] of the temporal sequence of I.

[4] A sequence $X = \langle x_1, x_2, ..., x_m \rangle$ is a subsequence of another sequence $Y = \langle y_1, y_2, ..., y_n \rangle$ if there exists a strictly increasing sequence $\langle i_1, i_2, ..., i_m \rangle$ of indices of Y such that for all $j = 1, 2, ..., m$, we

Proof: Let $S_G = <v_1, v_2, \ldots, v_n>$ and S_I be the temporal sequences of G and I, respectively. For any pair of vertices v_i and v_j in S_G where $i < j$, there are two possibilities on their temporal relationships: v_i is followed by v_j in G, and v_i and v_j are overlapped but v_i precedes v_j lexicographically. Since I supports G, it is clear that in either case the same relationship holds between v_i and v_j in I. Therefore, v_i appears before v_j in S_I. ÿ

Theorem 3.1 For a given clinical instance I, the traversal of the hash-tree examines every candidate temporal graph supported by I.

Proof: It is obvious that the traversal of the hash-tree for I visits the nodes by exhausting all the subsequences of the temporal sequence of I. According to Lemma 1, these nodes include all the candidate temporal graphs supported by I. ÿ

Use of a hash-tree at iteration k cannot only facilitate fast counting the support of candidate temporal graphs of size k but also improve efficiency of constructing the set of candidate temporal graphs of size k+1. Considering the hash-tree for candidate temporal graphs of size 3 shown in Figure 3.5, suppose that the temporal graph G_1 (i.e., $A \rightarrow B \rightarrow C$) has been found to be frequent. To find temporal graphs that are joinable with G_1, we need only to visit those leaf nodes that are reachable by hashing on B followed by C (assuming A is selected as the source vertex in G_1). In this case, both temporal graphs contained in the node 7 (i.e., G_4 and G_5) are joinable with G_1. This search process, of which correctness is ensured by Theorem 3.2, greatly reduces the overhead required to establish joinable temporal graphs for a given temporal graph.

---

have $x_j = y_{i_j}$ [CLR89].

Theorem 3.2 Let G be a frequent temporal graph with the temporal sequence being $<v_1, v_2, \ldots v_n>$. There must exist two joinable frequent temporal subgraphs G_1 and G_2 with temporal sequences being $<v_1, v_2, \ldots v_{n-1}>$ and $<v_2, v_3, \ldots v_n>$ respectively. In addition, the join set of G_1 and G_2 includes G.

Proof: It is obvious from Observation 3.1 that subtracting v_n from G results in a frequent temporal subgraph G_1 with the temporal sequence being $<v_1, v_2, \ldots v_{n-1}>$ and that subtracting v_1 from G results in a frequent temporal subgraph G_2 with a temporal sequence equal to $<v_2, v_3, \ldots v_n>$. Besides, since $G_2-\{v_n\} = G_1-\{v_1\}$, G_1 and G_2 are joinable, and their join set includes G. ÿ

3.3.2 TP-Itemset algorithm

TP-Itemset extends the Apriori algorithm for discovering patterns from a set of clinical instances. In TP-Itemset, each possible temporal relationship in a clinical instance I is explicitly represented as an item in the itemset for I. Each item in an itemset is of the form $v_i \rightarrow v_j$ if the activity v_i is followed by v_j or $v_i \sim v_j$ if the durations of activities v_i and v_j are temporally overlapped (where $\sim$ denotes an overlapping relationship and $v_i < v_j$ in their lexicographical order). For example, as shown in Figure 3.6, the clinical instance 1 is represented as $\{A \sim B, A \rightarrow C, B \rightarrow C\}$, while the clinical instance 2 is represented as $\{A \sim B, A \rightarrow C, B \sim C\}$. With this representation, $n(n-1)/2$ items are required to represent a clinical instance possessing n activities.

(a) Instance 1 (b) Instance 2

Figure 3.6 Examples of two clinical instances

As with the association rule discovery technique, an itemset that has certain user-specified minimum support is called a large itemset, while a potentially large itemset is called a candidate itemset. The TP-Itemset algorithm is similar to the Apriori algorithm but with one distinction. Unlike in the Apriori algorithm, where resulting large itemsets are unrestricted, a large itemset generated by the TP-Itemset algorithm needs to satisfy additional constraints. Let $V_S = \{v_1, v_2, \ldots, v_k\}$ be the set of distinct activities involved in an itemset *S*. An itemset *S* is referred to as a *full* itemset if the temporal relationship between any pair of activities, v_i and v_j where $v_i \in V_S$, $v_j \in V_S$, and $v_i \neq v_j$, exists in *S*. Otherwise, *S* is a *partial* itemset due to the absence of some temporal relationships in *S*. In effect, each full and large itemset corresponds with a frequent temporal graph defined in Definition 3.7. For instance, $\{A \sim B, A \rightarrow C, B \rightarrow C\}$ is a full itemset, while $\{A \sim B, A \rightarrow C\}$ is a partial itemset because the relationship between activities *B* and *C* is unspecified. Hence, large itemsets generated by the TP-Itemset algorithm are required to be full itemsets.

Given a set of clinical instances, the TP-Itemset algorithm transforms them into a set of full itemsets and generates the itemsets among all full and large itemsets. Each such full and large itemset represents a structure pattern. Let L_k be a set of large *k*-itemsets each of which has *k* items (i.e., *k* temporal relationships) and C_k be a set of candidate *k*-itemsets, where C_k can be constructed by joining large itemsets in L_{k-1}. In the Apriori algorithm, the joining procedure requires that items within an itemset be kept in their lexicographic order. In this study, since each item ($v_i \rightarrow v_j$ or $v_i \sim v_j$) involves two activities, the lexicographical order of a set of items is based on their first activities (i.e., v_i) and then on their second ones (i.e., v_j). Moreover, the partial itemsets from L_{k-1} should not be removed immediately at each iteration *k*–1, since two

partial $(k-1)$-itemsets may result in a full itemset in C_k. For instance, joining two partial, large itemsets in L_2, $\{A \rightarrow B, A \rightarrow C\}$ and $\{A \rightarrow C, B \sim C\}$, results in a full itemset $\{A \rightarrow B, A \rightarrow C, B \sim C\}$. Finally, to facilitate fast counting the support for the candidate itemsets in C_k, candidate itemsets are stored in a hash-tree as employed by the Apriori algorithm [AS94]. Accordingly, the TP-Itemset algorithm for discovering structure patterns is listed below.

TP-Itemset(a set of process instances: *I*, the minimum support: *minsup*): a set of large and full itemsets
{
Transform each clinical instance $i \in I$ into a itemset s in S;
L_1 = {large 1-itemsets};
$MaxK$= 1;
For (k = 2; $L_{k-1} \neq \emptyset$; k++)
{
Generates candidate itemsets C_k from L_{k-1};
For each itemset $s \in S$
{
C_t = subset[5](C_k, i); // find candidate itemsets that are supported by i
For each candidate $c \in C_t$ do c.count++;
} /* end-for */
$L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$;
If ($L_k \neq \emptyset$) then $MaxK = MaxK+1$;
} /* end-for */
For (k=$MaxK$; k>1; k--)
{
Prunes partial itemsets in L_k;
} /* end-for */
Return $\cup_{k \geq 1} L_k$;
}

3.3.3 TP-Sequence algorithm

TP-Sequence is based on the sequential pattern discovery technique (specifically, the AprioriAll algorithm) to discover patterns from a set of clinical instances. In the TP-Sequence algorithm, the overlapped and followed relationships in each clinical

[5] Using the hash-tree constructed for C_k, the subset(C_k, i) function is to find all the candidate itemsets in C_k that are supported by the itemset i. We employed and implemented the subset function as proposed in [AS94].

instance are explicitly represented as a sequence, where an itemset is a non-empty set of overlapping activities, and a sequence is an ordered list of itemsets. The itemset (x, y) denotes that activities x and y are temporally overlapped. A sequence with an order list of k itemsets is called a k-sequence. For instance, a 2-sequence <(x)(y)> denotes that the activity x is followed by y. Furthermore, a 2-sequence <(x, y)(y, z)> denotes that activity x is followed by z, while y overlaps with x and z. As shown in Figure 3.6, the clinical instance 1 is represented as a sequence of <(A, B) (C)>, while the clinical instance 2 is represented as <(A, B) (B, C)>. Using this representation, a clinical instance is represented as an m-sequence where $l \leq m \leq n$, l is the number of activities in the longest path in the respective temporal graph, and n is the number of activities in the clinical instance.

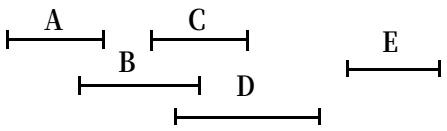
In the sequential pattern discovery, no specific constraint is imposed on itemsets in a sequence. However, since a sequence in the TP-Sequence algorithm is used to represent both the followed and overlapped relationships, a meaningful sequence needs to satisfy certain constraints. We call a sequence <(x) (x, y)> a *non-canonical* sequence since it is identical to <(x, y)>. On the other hand, <(x, y) (w) (x, z)> is an *illegitimate* sequence since both "x followed by w" and "w followed by x" exist; thus, violating the irreflexivity of followed relationships. Hence, for discovering structure patterns, the AprioriAll algorithm needs to be extended to ensure that any candidate sequence generated be canonical and legitimate. Let a_i be an itemset. Non-canonical and illegitimate sequences are formally defined as follows.

Definition 3.11 A sequence $s = <a_1\ a_2\ \ldots a_m>$ is canonical if for each itemset a_j, $1 \leq j < m$, $a_j \not\subset a_{j+1}$ and $a_{j+1} \not\subset a_j$.

Definition 3.12 A sequence $s = <a_1\ a_2\ \ldots a_m>$ is *legitimate* if for each item x

involved in *s*, the itemsets that contain *x* form a continuous sequence in *s*.

To distinguish an unconstrained sequence from a sequence with canonicity and legitimacy properties (as required by the TP-Sequence algorithm), the latter is called a *quasi-sequence*. The transformation of a clinical instance *I* into its respective quasi-sequence proceeds in the following iterative manner. The quasi-sequence is initialized as an empty list. We traverse the starting and ending times of activities in *I* in ascending order. The set *O* of overlapped activities is accumulated each time a starting time is visited. When the first ending time is encountered, the set *O* is appended to the quasi-sequence. Subsequently, we continue the traversal until the next starting time (assuming its respective activity be *v*) is visited. The subset of activities in *O* whose ending times appear before the starting time of *v* are removed from *O* since this subset of activities that have appeared in the quasi-sequence are followed by *v*. This traversal procedure continues until all the timestamps are visited. Let us illustrate this transformation using the clinical temporal grap instance shown in Figure 3.7(a). As shown, when the first ending time (which belongs to *A*) is visited, *O*=(*A*, *B*) and, thus, the current quasi-sequence is <(*A*, *B*)>. Since there no ending times appear before the next starting time (that pertains to *C*), only *A* is removed. When the next ending time (that pertains to *B*) is visited, *O*=(*B*, *C*, *D*) and, therefore, the quasi-sequence becomes <(*A*, *B*) (*B*, *C*, *D*)>. When the following starting time (that is possessed by *E*) is visited, *O* becomes empty because the ending times of *B*, *C*, and *D* have all been traversed. When the last ending time (which belongs to *E*) is visited, *O*=(*E*), and the resultant quasi-sequence is <(*A*, *B*) (*B*, *C*, *D*) (*E*)> as shown in Figure 3.7(b). The pseudo-code of the described transformation is listed in the following.



<(A,B)(B,C,D)(E)>

(a) An instance (b) Respective quasi-sequence

Figure 3.7 Examples of clinical instance and quasi-sequence

```
Generate-Quasi-Sequence(a clinical instance: I): a quasi-sequence
{
    Quasi-Seq = <>; /* Initialization of a quasi-sequence */
    Sort the timestamps of activities in I and place them in a queue time;
    O = Ø;
    While time ≠ null {
        if time.event = ' starting time' {
            add time.activity to O;
            time = time.next;
        }
        else {/* time.event = ' ending time' */
            Insert O to Quasi-Seq;
            While (time.next ≠ null) and (time.event = ' ending time' ) {
                O= O - time.activity;
                time = time.next;
            } /* end-while */
        } /* end-if */
    } /*end-while */
    Return Quasi-Seq;
}
```

In a quasi-sequence, when an activity v appears in two consecutive itemsets I_j and I_{j+1}, v is temporally overlapped with the remaining activities in I_j and I_{j+1}, rather than v in I_j taking place before the activities in I_{j+1} and v in I_{j+1} occurring after the activities in I_j. This unique interpretation requires re-definition of the subsequence relationship used by the sequential pattern discovery algorithm. Using the example shown in Figure 3.7, the clinical instance is represented as a 3-quasi-sequence <(*A*, *B*) (*B*, *C*, *D*) (*E*)>. In the sequential pattern discovery, the sequence <(*B*) (*C*)> is considered to be supported by (or a subsequence of) <(*A*, *B*) (*B*, *C*, *D*) (*E*)>. However, the quasi-sequence <(*B*) (*C*)> indeed denotes a followed relationship between *B* and *C*, which differs from an overlapped relationship in the clinical instance under discussion. Thus, the quasi-sequence <(*B*) (*C*)> is not supported by <(*A*, *B*) (*B*, *C*, *D*) (*E*)> in the structure

pattern discovery. The re-defined subsequence relationship (formally defined in Definition 3.13) is needed when determining support for candidate quasi-sequences.

Definition 3.13 A quasi-sequence $s = <a_1\ a_2\ \ldots a_n>$ is supported by (or *a subsequence of*) another quasi-sequence $t = <b_1\ b_2\ \ldots b_m>$, if there exists integers $i_1 < i_2 < \ldots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$, and there exists no consecutive itemsets a_j and a_{j+1} in s such that $v \in a_i$, $u \in a_{i+1}$, and the itemset $(u, v) \subseteq b_{ij}$ or $(u, v) \subseteq b_{ij+1}$.

Accordingly, the TP-Sequence algorithm, extending the AprioriAll algorithm, finds all frequent quasi-sequences. Each such quasi-sequence corresponds with a structure pattern. The TP-Sequence algorithm employs the same hash-tree data structure as the AprioriAll algorithm for storing candidate quasi-sequences in C_k and fast counting their support [AS95]. The pseudo-code for the proposed TP-Sequence algorithm is listed as follows.

```
TP-Sequence (a set of clinical instances: TIS, the minimum support: minsup): a set of large quasi-sequences
{
    QSS = Ø; /* QSS contains a set of quasi-sequences for instances */
    For each clinical instance I in TIS
    {
        QSS = QSS ∪ Generate-Quasi-Sequence(I);
    } /* end-for */
    L1 = {large 1-quasi-sequences};
    For (k = 2; Lk-1 ≠ Ø; k++)
    {
        Ck = candidate sequences generated from Lk-1;
        Delete non-canonical and illegitimate sequences in Ck;
        For each quasi-sequence qs in QSS
        {
            Increment the count of all candidates in Ck that are supported by qs;
        } /* end-for */
        Lk = {c ∈ Ck | c.count ≥ minsup};
    } /*end-for */
    Return ∪k≥1Lk;
}
```

3.4 Performance evaluation

In this section, we evaluate the performance and scale-up properties of the three proposed algorithms for finding structure patterns. The experiments were conducted on an IBM compatible PC with a CPU clock rate of 500 MHz and 128 MB of main memory, running FreeBSD 4.1. Since we intend to examine the performance and scalability of the proposed algorithms over a wide range of data characteristics, synthetic data sets were generated and employed for the evaluation.

3.4.1 Generation of synthetic data

To generate synthetic data set for clinical instances, we adopted and extended the transaction generation model proposed in [AS95] for evaluating the Apriori and AprioriAll algorithms. In our model of executions, clinical instances are not randomly designed but tend to contain sets of temporally related activities, each of which is a potential structure pattern. Furthermore, clinical instances are generated based on these potential structure patterns. However, a clinical instance may include only a subset of activities from a potential structure pattern.

Given a set A of available activities with size N, we first generate a pool of potential structure patterns. The number of such patterns generated is set to PN. A potential structure pattern P is generated by first determining its size (i.e., the number of activities) from a Poisson distribution with mean equal to PS. Activities in the first potential structure pattern are chosen randomly. To model the phenomenon where structure patterns may involve common activities, some percentage of activities in P, determined by an exponentially distributed random variable with mean equal to the correlation ratio (CR), are randomly chosen from the potential structure pattern Q

generated immediately prior to *P*. Subsequently, the remaining activities in *P* are selected randomly from the rest of activities in *A*. In addition, to determine temporal relationships among those activities in *P*, some fraction of activities are chosen and arranged in sequence; thus exhibiting followed relationships. We use an exponentially distributed random variable with mean equal to the length ratio (*LR* defined as the maximal number of sequential activities to the total number of activities) to decide this faction for each pattern. Furthermore, without loss of generality, we assume the execution duration of each such sequential activity in *P* be identical. For each remaining activity in *P*, its execution interval e_i, bounded by the earliest starting time and the latest ending time of the sequential activities decided previously, is randomly determined, thus creating overlapped relationships. However, e_i should not reside in the time gap between any two consecutive sequential activities in order to preserve the pre-decided length ratio for *P*.

After the generation of the set of potential structure patterns, each pattern is assigned a weight, which corresponds to its probability of being selected when generating a clinical instance. The weight is initially picked from an exponential distribution with unit mean and then normalized so that the sum of the weights for all of the patterns is 1. Finally, the set of clinical instances are generated. The size of a clinical instance is picked from a Poisson distribution with mean equal to *IS*. For each clinical instance, one of the potential structure patterns is randomly chosen by tossing a *PN*-sided weighted coin, where the weight for a side is the probability of picking the associated pattern. If the size of the chosen pattern is not the same as that of the instance, surplus activities are randomly dropped or additional activities are randomly added in an overlapped manner.

The parameters and their respective default values used for the generation of synthetic data sets are summarized in Table 3.1. Depending on the type of experiments conducted, the respective parameter will be examined over a range of values, while the rest of parameters adopt their default values. For each particular experiment, 10 trials were performed and the overall performance was then estimated by averaging the performance across all trials.

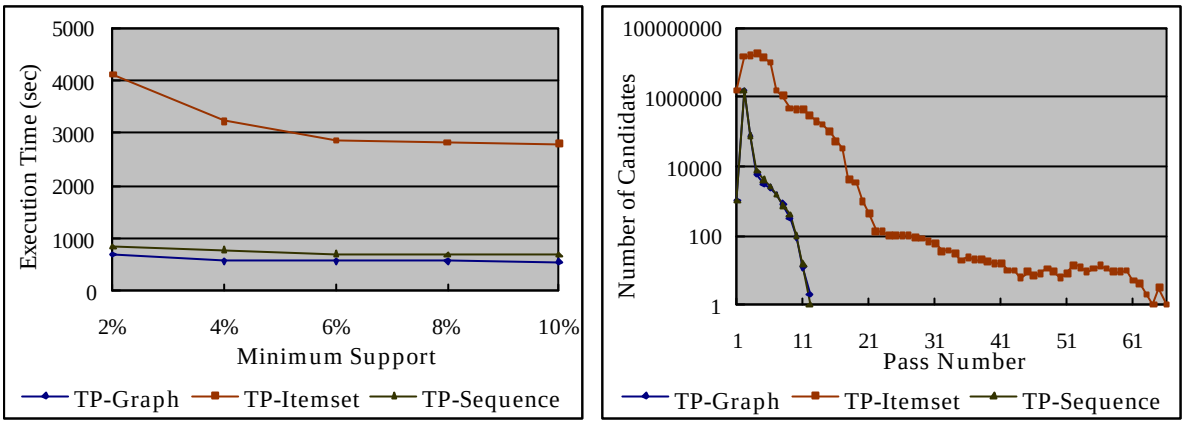
Table 3.1 Parameters and default values for synthetic data generation

Symbol	Description	Default
N	Number of activities	1,000
D	Number of clinical instances	10,000
IS	Size of clinical instances	20
PN	Number of potential structure patterns	1,000
PS	Size of potential structure patterns	10
CR	Correlation ratio	0.5
LR	Length ratio	0.5

3.4.2 Effects of minimum support thresholds

Ten synthetic data sets were generated using the default values for all parameters as depicted in Table 3.1. We investigated the effects of minimum support thresholds, ranging from 2% to 10% at 2% increments, on the execution times of each proposed algorithm. Figure 3.8(a) shows the execution times of the three proposed algorithms as a function of minimum support. As expected, the execution times of the three algorithms decreased as the minimum support increased. Over the range of minimum supports investigated, a decrease in minimum support appeared to have shown marginal effects on the execution times of TP-Graph and TP-Sequence. On the other hand, a decrease in minimum support resulted in a noticeable increase in the execution times of TP-Itemset. Across all the minimum supports examined, TP-Graph appeared to have exhibited the best performance, while TP-Itemset performed worst,

mainly because it generated and counted a much larger number of candidates than the other two algorithms. As shown in Figure 3.8(b), when the minimum support was 2%, the number of candidates generated and the number of iterations (i.e., passes) taken by TP-Itemset were significantly higher than those produced/required by its counterparts. Such dramatic differences could be attributed to their underlying structures for representing and manipulating clinical instances and candidates. TP-Itemset explicitly represents each temporal relationship (followed or overlapped) as an item in an itemset and generates candidates at the temporal relationship level. Thus, the size of C_1 (i.e., containing all possible relationships between pairs of activities) considered by TP-Itemset is $3n(n-1)/2$, where n is the number of activities, leading to even larger candidate sets in the first few passes. TP-Graph and TP-Sequence forms a candidate temporal graph and quasi-sequence, respectively, by adding an additional activity from the previous iteration. Hence, the number of activities considered in pass 1 by either algorithm is n, which is far fewer than those generated by the TP-Itemset when n is large. On the other hand, assuming the maximal number of activities in the structure patterns to discover to be s, the number of passes required by TP-Graph and TP-Sequence is at most $s+1$. However, the number of passes for generating and counting candidate itemsets would be $s(s-1)/2$ or higher. The larger candidate sets and higher number of passes considered by TP-Itemset resulted in its inferior performance. Regarding the performance of TP-Graph and TP-Sequence, which similarly represent and manipulate clinical instances and candidates, generated almost identical number of candidates at all iterations and required the same number of passes for the target structure pattern discovery, leading to comparable performances measured by execution time. However, a more concise representation of clinical instances and structure patterns employed in TP-Graph appeared to contribute to its better performance than TP-Sequence.



(a) Effects of Minimum Supports (b) Size of Candidates (Minimum Support = 2%)

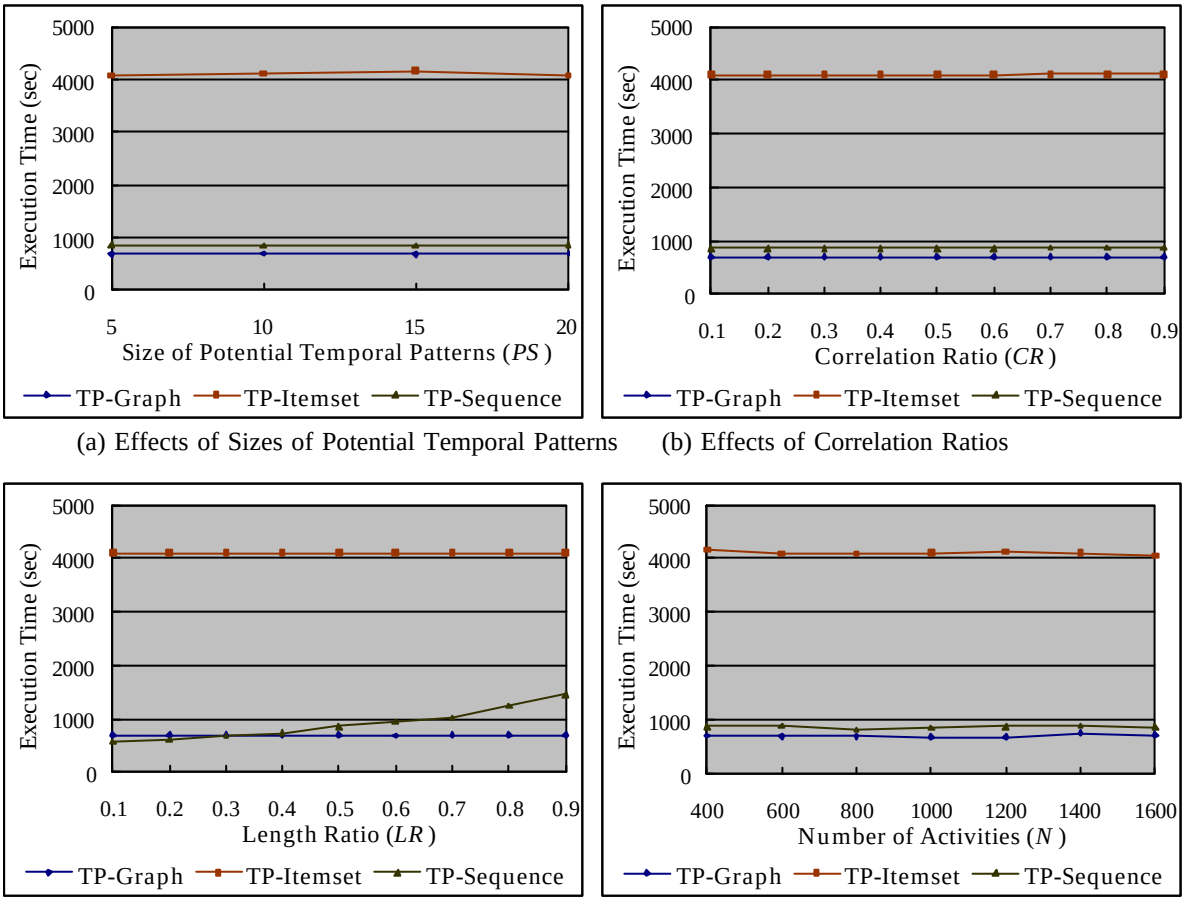
Figure 3.8 Experimental results: effects of minimum support thresholds

3.4.3 Effects of instance characteristics

The performances of the three algorithms were evaluated over a range of instance characteristics described by size of potential structure patterns (*PS*), correlation ratio (*CR*), length ratio (*LR*), and number of activities (*N*) available for generating potential structure patterns and clinical instances. We did not examine the effects of number of potential structure patterns (*PN*) since varying the value of *PN* is similar to adjusting the minimum support threshold for a given value of *PN*.

Synthetic data sets were generated for various sizes of potential structure patterns, ranging from 5 to 20 at 5 increments. Remaining parameters received their default values, as defined in Table 1. The minimum support was set to 2%. Figure 3.9(a) shows the execution times of the three algorithms as functions of the size of potential structure patterns. The performance of the three algorithms remained largely stable across the sizes of potential structure patterns examined. The resulting execution times of TP-Graph, relatively comparable to those of TP-Sequence, were significantly lower than those required by TP-Itemset. Various correlation ratios, ranging from 0.1 to 0.9 at 0.1 increments were also investigated. At a minimum support of 2%, a steady

performance was achieved by all of the proposed algorithms across all correlation ratios examined, as shown in Figure 3.9(b). As with the previous experiment, TP-Graph was relatively comparable to TP-Sequence and outperformed TP-Itemset.



(a) Effects of Sizes of Potential Temporal Patterns (b) Effects of Correlation Ratios

(c) Effects of Length Ratios (d) Effects of Numbers of Activities

Figure 3.9 Experimental results: effects of instance characteristics

In addition, we investigated the effects of length ratios, ranging from 0.1 to 0.9 at 0.1 increments, on the performance of the three algorithms. As shown in Figure 3.9(c), at a minimum support of 2%, the execution times of TP-Graph and TP-Itemset remained stable across different levels of length ratio examined. However, the execution times attained by TP-Sequence increased linearly as the length ratio grew from 0.1 to 0.9. A larger length ratio represents a scenario in which potential structure patterns and their respected clinical instances were more likely to contain sequential activities; thus

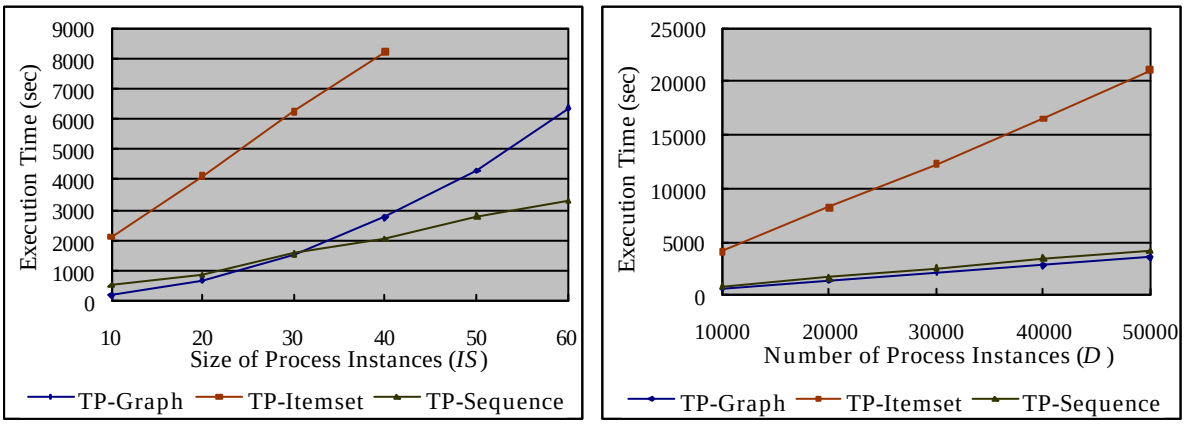
requiring a longer quasi-sequence for representing each clinical instance. As a result, as length ratio increased, the number of passes for generating and counting candidate quasi-sequences increased and the performance of TP-Sequence degraded. Conversely, given the same set of activities appearing in a clinical instance, an increase in its length ratio did not increase the size of the resulting temporal graph or itemset. Thus, length ratio appeared to have no effect on the execution times of TP-Graph and TP-Itemset. Overall, TP-Graph was the most efficient algorithm, followed by TP-Sequence and finally TP-Itemset.

Finally, the effects of the numbers of activities (ranging from 400 to 1600 at increments of 200) available for generating potential structure patterns and the clinical instances on the performance of the three algorithms were examined. The minimum support was again set to 2%. As shown in Figure 9(d), the performance of the three algorithms remained largely stable across different numbers of activities examined. The execution times needed by TP-Graph, largely comparable to those by TP-Sequence, were significantly lower than those attained by TP-Itemset.

3.4.4 Scale-up experiments

The scalability experiments in this study were designed from two different perspectives: (1) by increasing the average size of clinical instances (*IS*) while keeping the number of instances constant and (2) by increasing the number of instances (*D*) while keeping the average size of instances constant. The first scale-up experiment increased the average size of instances, ranging from 10 to 60 at 10 increments. The remaining parameters received their default values as depicted in Table 1. Figure 3.10(a) shows the execution times required by TP-Graph, TP-Itemset

and TP-Sequence, respectively, at a minimum support of 2%. We have not plotted the execution times for TP-Itemset when the size of clinical instances was greater than 40, since TP-Itemset generated too many candidates and ran out of memory. When the size of clinical instances increased from 10 to 40, the execution time of TP-Itemset increased linearly. However, the execution times of TP-Graph appeared to scale fairly quadratically across the range of sizes of instances examined. Because each clinical instance is represented as a graph that requires quadratic manipulation, the execution time increases toward a quadratic trend as the size of instances expands linearly. On the other hand, the execution times of TP-Sequence appeared to scale linearly. Such linear performance with respect to the size of instances can be attributed to the linear manipulation of sequences of itemsets. When the size of instances was below 30, TP-Graph was the most efficient algorithm. However, as the size of instances exceeded 30, TP-Sequence exhibited better performance.



(a) Effects of Sizes of Clinical Instances (*IS*) (b) Effects of Numbers of Clinical Instances

Figure 3.10 Results of scale-up experiments

The second scale-up experiment varied the number of instances (*D*), ranging from 10,000 to 50,000 at increments of 10,000, while adopting their default values for the remaining parameters. Figure 3.10(b) shows the performances of the proposed

algorithms as a function of the number of instances, at a minimum support of 2%. As shown, all of the proposed algorithms grew linearly with the number of instances. The increasing rate for TP-Graph appeared to be the smallest, while TP-Itemset exhibited the worst performance and scalability with respect to the number of instances.

3.5 Summary

In this chapter, we formally defined the structure pattern discovery problem, and developed and evaluated three different algorithms, namely TP-Graph, TP-Itemset and TP-Sequence, for finding a set of temporal structure patterns from process instances.

Using synthetic data sets, we analyzed the performance, over a range of data characteristics, and scale-up properties of the three proposed algorithms. The experimental results showed that the size of potential temporal patterns, correlation ratio, length ratio and the number of available activities had no, or at most marginal, effects on the execution times of the proposed algorithms. Overall, TP-Graph appeared to achieve the best performance. Due to its representation and manipulation that treat each temporal relationship in a clinical instance as an individual item, TP-Itemset exhibited the worst performance. In terms of scale-up properties, the experimental results suggested that the execution times of TP-Sequence and TP-Itemset grew linearly as the size of clinical instances expanded linearly, while those of TP-Graph increased toward a quadratic growth. The experimental results also suggested that the three proposed algorithms scaled linearly with the number of clinical instances, with the TP-Graph algorithm achieving the best scalability.

Chapter 4
Feature selection

In our induction work, frequent structure patterns discovered by algorithms described in the previous chapter are regarded as features. Therefore, we are given a training set of labeled fixed-length feature vectors, or instances, from which to learn an induction model. This model is then used to predict the classes (or labels)– normal or fraudulent – for a set of unlabeled instances. Thus, the information about the classes that inherent in features determines the accuracy of the model. Theoretically, having more features should give us more discriminating power. However, the real world provides us with many reasons why this theoretical observation does not hold in practice.

First, it is well recognized that the number of features has a strong impact on the performance of an induction algorithm. The time requirements for an induction algorithm often grow dramatically with the number of features, rendering the algorithm impractical for problems with a large number of features. In our case, applying real insurance data to a structure pattern discovery algorithm generates more than 10000 structure patterns. Therefore, it is imperative to reduce the feature set prior to constructing a classifier since the computational complexity of induction algorithms depends heavily on the number of features.

Furthermore, many induction algorithms, such as decision trees [Quinlan93] and Bayes classifier [DH73], can be viewed as performing estimation of the conditional probability of the class label given a set of features. Irrelevant and redundant features cause problems as they may confuse the induction algorithm by helping to obscure the

distributions of the small set of truly relevant features. The inclusion of such irrelevant or redundant features in the training data may in turn degrade the accuracy of an induction algorithm.

In summary, reducing the set of features before conducting the induction algorithm serves two purposes: to decrease the running time of the induction algorithm and to increase the accuracy of the resulting model. We therefore address this feature selection issue in this chapter. We first give an overview of related works, and present in-depth how to select relevant features to construct a detection classifier. We finally describe procedure and results of experiments in detecting fraudulent instances.

4.1 Related works

A number of researches have addressed the problem of feature subset selection. As noted by [JKP94], this work is often accomplished along two different lines: wrapper and filter models.

The wrapper model [JKP94][CF94][LS94][BL97] scans through the space of feature subsets in search of the one that has the highest estimated accuracy from an induction algorithm. Thus, the feature selection sits on top of an induction algorithm, and the feature subset search and the underlying induction algorithm strongly interact. While these methods have been shown to achieve some success on induction, they suffer from high computation cost and are not applicable to tasks with only a few hundred features.

The filter model introduces a preprocessing step prior to induction. As such, the

adoption of the induction algorithm does not interfere with the selection of the feature selection algorithm. A major benefit with the filter model is that it does not need to search through the space of feature subsets as in the wrapper models, and is therefore efficient for domains containing a large number of features.

Three of the most well-known filter methods for feature selection are RELIEF [KR92], FOCUS [AD94], and Markov blanket filter [KS96]. In RELIEF, each feature is individually assigned a weight indicating its relevance to the class label, and a subset of features with high weights is selected. RELIEF therefore may fail to remove redundant features as two predictive, but highly correlated, features will both be selected.

The FOCUS algorithm exhaustively searches all feature subsets in order to identify a minimal set of features that consistently label instances in the training data. This consistency criterion makes FOCUS vulnerable to noise in the training data. Moreover, searching the power set of the features also makes this algorithm impractical for domains with a large number of features.

Koller and Sahami develop a probability framework, called Markov blanket filter, for selecting an optimal subset of features [KS96]. Theoretically, this method eliminates a feature if it gives no additional information beyond that subsumed by a subset of the remaining features (called Markov blanket). Since finding Markov blanket of a feature might be computational infeasible, this research induces an algorithm that computes an approximation to the optimal feature set.

In our domain, typically with a large number of features, filter model has the key

advantage of computation cost. For this reason, we focus our attention on filter model. We address both theoretical and empirical aspects of feature selection with respect to the classification task. We describe a formal framework for understanding feature selection in our domain and present efficient algorithms based on these theoretical intuitions in next sections.

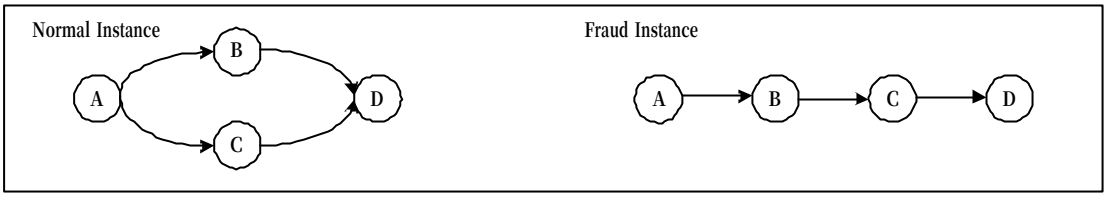
4.2 Formalization of feature selection problem

In our classification task, patterns discovered by mining algorithms are regarded as features, each of which denotes whether a specific pattern is supported by an instance. Thus, each instance can be translated as a set of feature values and a class label. Our view on a translated example can be formally described as below.

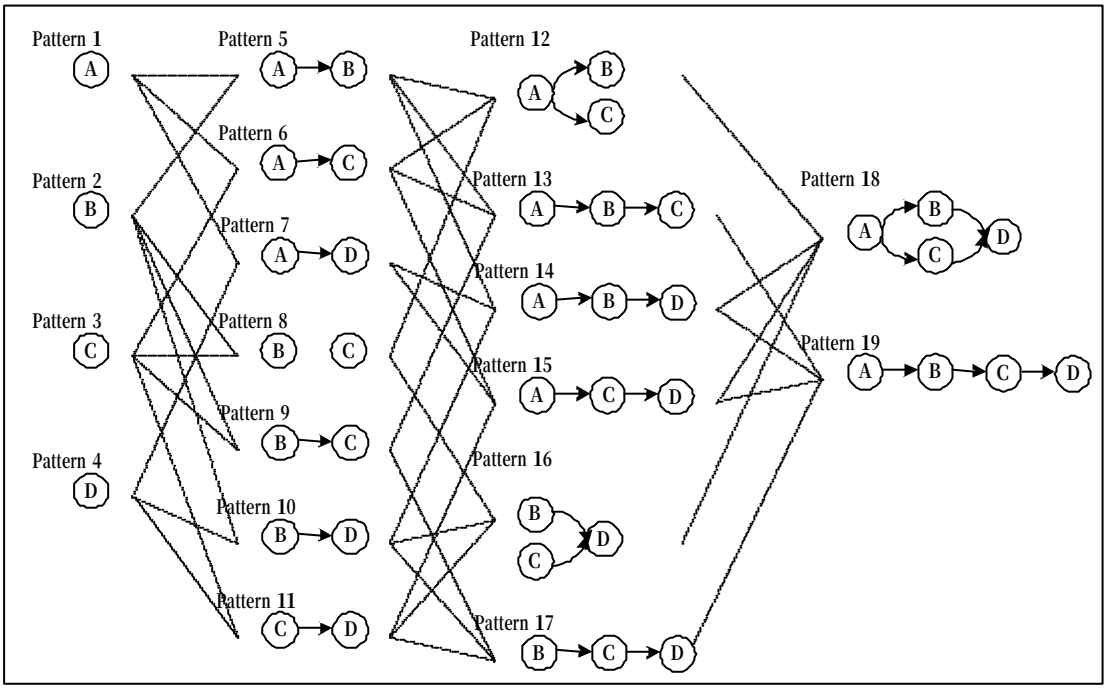
Definition 4.1 A translated example *IE* of an instance *I* is a pair (***f***, *c*), where ***f*** and *c* are a set of feature values and a class label of *IE* respectively. $\boldsymbol{f} = (f_1, f_2, \ldots, f_n)$ is an assignment of a set of Boolean features $\boldsymbol{F} = (F_1, F_2, \ldots, F_n)$, in which feature f_j is set to 1 if and only if instance *I* supports the corresponding pattern of F_j. *c* is an assignment of a binary variable *C*, which is set to 1 if and only if *I* is a fraudulent instance.

Consider an example shown in Figure 4.1. Suppose there are two instances, which are gauged as normal and fraudulent in our data set, as shown in Figure 4.1(a). With 50% support threshold, 19 patterns, shown in Figure 4.1(b), are discovered by mining algorithms. Therefore, two translated examples, shown in Figure 4.1(c), are generated, each having 19 corresponding feature values and one class label. In the first translated example, 1 is assigned to Features 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 14, 15, 16, and 18

since the first instance supports 15 corresponding patterns and 0 is assigned to the other features. Similarly in the second example, Features 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 13, 14, 15, 17, are assigned 1, and the other features are assigned 0. Besides, the first example is labeled 0, while the second example is labeled 1.



(a) Two instances



(b) Discovered patterns

	Feature																			Class
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
Translated example	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	0	0
	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	0	1	1

(c) Two translated examples

Figure 4.1 Instances and their translated examples

By Definition 4.1, a detection classifier is a procedure that takes as input a translated

example *IE* which is a feature assignment ***f***. The classifier predicts that the instance is a member of one of the 2 possible classes {0, 1}. The classifier must make its decision based on the assignment ***f*** associated with example *IE*.

In theory, the feature variables will fully determine the appropriate classification. However, this is rarely the case in practice since we do not usually have access to enough features to make a deterministic decision. Therefore, we use a probability distribution to model the classification function. For each feature assignment ***f***, we have a distribution Pr(*C* | ***F*** = ***f***), where *C* is the class random variable. An induction algorithm implicitly uses the empirical frequencies observed in the training set–an approximation to the conditional distribution Pr(*C* | ***F***) –to construct a classifier for our detection problem.

Let us consider the effect of feature space reduction on the distribution that characterizes the problem. Let ***F'*** be some subset of ***F***. Given an assignment ***f***, we use ***f'*** to denote the projection of ***f*** onto the variables in ***F'***. Consider a particular example characterized by ***f***. In the original distribution, this example induces the distribution Pr(*C* | ***F*** = ***f***). In the reduced feature space, the same example induces the (possibly different distribution) Pr(*C* | ***F'***= ***f'***). Our goal is to select ***F'*** so that these two distributions are as close as possible.

To compute the difference between two distributions, we use the information-theoretic measure of cross entropy [KL51] given in Eq 4.1, where *a* and *b* denote two distinct distributions. While other measures of difference between two distributions (notably divergence) have been suggested in the statistics community [Fukunaga90], they are often aimed at selecting features to enhance the separability of the data, which is

difficult to achieve in an environment with very large dimensional spaces. Hence, we use cross entropy as our distance metric, and view the process of selecting features as selecting a set of features ***F'*** which causes us to lose the least amount of information in the distribution.

$$D(a,b) = \sum_{x \in \Omega} a(x) \log \frac{a(x)}{b(x)} \quad \text{.....(Eq 4.1)}$$

Cross entropy between two distributions a and b, denoted as $D(a, b)$, measures the extent of the "error" made by using b as an approximation to a. Thus, this measure is particularly suitable for our application, with $\Pr(C \mid \boldsymbol{F})$ playing the role of the "more informed" distribution a, and $\Pr(C|\boldsymbol{F'})$ playing the role of an approximation distribution b. In this case, the probability space Ω is the set of possible classes $\{0, 1\}$.

Besides, we need to aggregate the cross entropy values for different feature vectors $\boldsymbol{f}$ into a single quantity. Naively, we might think to simply sum their cross entropy values, or to consider just the maximum cross entropy value. Neither ideas take into consideration the fact that some feature vectors are far more likely to occur than others, and that we might not mind making a larger mistake in certain rare cases. Therefore, we use a weighted measure [KS96] shown in Eq 4.2 to aggregate cross entropy values for a given feature subset $\boldsymbol{F'}$, where $\Pr(\boldsymbol{f})$ denotes the probability of a vector $\boldsymbol{f}$. Clearly, our goal is to select a subset $\boldsymbol{F'}$ of features so that the measure $\Delta_{F'}$ is minimized.

$$\Delta_{F'} = \sum_{f} \Pr(f) \times D(\Pr(C \mid f), \Pr(C \mid f')) \quad \text{.....(Eq 4.2)}$$

Problem statement: Given a set of translated examples, each of which is represented as a set of feature values ***f*** and a class label *c*, the feature selection problem is to find a subset ***F'*** of ***F*** so that $\Delta_{F'}$ is minimized.

4.3 Feature selection algorithms

From the above problem definition, we can easily conclude that the feature set that minimizes $\Delta_{F'}$ is simply ***F***, since it maintains the exact distribution. This observation might suggest that we can use a backward elimination algorithm, where at each state a feature F_i is eliminated in a way that allows us to remain as close to the distribution as possible. That is, we have a current feature set ***F'***, initially set to ***F***. At each stage, we eliminate a feature F_i such that $\Delta_{F'-F_i}$ is as close as possible to $\Delta_{F'}$.

Unfortunately, it is impractical to simply implement this greedy procedure for several reasons. First, the computation time of $\Delta_{F'}$ is exponential to the number of features. Second, a greedy algorithm may result in less than optimal solution. Furthermore, we cannot really compare our approximate distribution to the true conditional distribution Pr(C| ***F***), since the true distribution is not available to us. Rather, we have a training set which provides us with only a rough approximation to it. In our case, we have a large number of features, and the number of examples in the training set corresponding to any particular assignment ***f*** will be very small or even non-existent. Therefore, as the number of features grows, our ability to use the training set to approximate this conditional distribution decreases (exponentially).

Therefore, we utilize ideas from probabilistic reasoning to circumvent this problem. Intuitively, features that cause a small difference of distributions are those that give us

the least additional information beyond what we would obtain from the other features. We can capture this intuition via the formal notion of *conditional independence* defined as below.

Definition 4.2 [Pearl88] Tow sets of variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ are said to be *conditionally independent* given some set of variables $\boldsymbol{Z}$ if, for any assignment values of $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$, to the variables $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ respectively, $\Pr(\boldsymbol{X}=\boldsymbol{x} \mid \boldsymbol{Y}=\boldsymbol{y}, \boldsymbol{Z}=\boldsymbol{z}) = \Pr(\boldsymbol{X}=\boldsymbol{x} \mid \boldsymbol{Z}=\boldsymbol{z})$. That is, $\boldsymbol{Y}$ gives us no information about $\boldsymbol{X}$ beyond what is already in $\boldsymbol{Z}$.

Thus, we can eliminate a conditionally independent feature F_i without increasing the distance from the desired distribution. While it is also impractical to test for conditional independence, this idea sheds light to a solution. As we will show, we exploit sub-/super- relationships among discovered patterns to efficiently eliminate features.

Definition 4.3 Suppose F_i corresponds to a k-sized structure pattern i. A feature F_j, corresponding to structure pattern j, is said to be a *descendant* of F_i if j is a temporal subgraph of i. A descendant of F_i is also a *child* of F_i if it is of size $k-1$. The set of children of F_i is denoted as Child(F_i), and the set of descendants of F_i is denoted as Descendant(F_i).

Definition 4.4 A feature F_j is said to be a *parent* of F_i if F_i is a child of F_j. A feature F_j is said to be an ancestor of F_i if F_i is a descendant of F_j. We use Parent(F_i) and Ancestor(F_i) to denote the set of F_i's parents and the set of F_i's ancestors respectively.

Take Figure 4.1 as an example. Feature 1, 2, 3, 4, 5, 6, and 8, obviously, are the

descendants of Feature 12, since their corresponding Pattern 1, 2, 3, 4, 5, 6, and 8 are temporal subgraphs of Pattern 12. Feature 5, 6, and 8 are the children of Feature 12. Similarly, Feature 18 is the ancestor of Feature 12 since the corresponding pattern of Feature 18 is indeed a temporal supergraph of that of Pattern 12.

Recall from Chapter 3 that the downward closure property states that if a pattern i has support of at least s%, any temporal subgraph of i must have a support of at least s%. Let X be a set of instances, each supporting all temporal subgraphs of pattern i and Y be a set of instances that support pattern i. Obviously $X \supseteq Y$. Therefore, if X falls into the category of a particular class, Y must belong to the same class. From the classification point of view, F_i thus give us no further information than that provided by F_i's children. In the corresponding training set, we can give a lemma as below.

Lemma 4.1. Suppose $\boldsymbol{A} \subseteq$ Descendant(F_i) is a set of features. Let $\boldsymbol{E}$ be the set of translated examples that have value 1's in every feature of $\boldsymbol{A}$. If every translated example in $\boldsymbol{E}$ has the same class label $c_l \in C$, then F_i and C are conditionally independent given $\boldsymbol{A}$.

Proof:

We need to prove Pr($C|\boldsymbol{A}$, F_i) = Pr($C|\boldsymbol{A}$). To do so, we divide the instances into two partitions P_1 and P_2:

(1) P_1: Each instance in P_1 has 0 in at least a feature value in $\boldsymbol{A}$. Let a_1 be a feature value A in P_1.
Clearly, the F_i value of each instance in P_1 must be 0. Therefore, in this case, we have
$\Pr(C \mid A = a_1) = \Pr(C \mid A = a_1, F_i = 0)$.

(2) P_2: Each instance in P_2 has 1 in every feature value in $\boldsymbol{A}$. Let a_2 be such the feature value in A.
In this case, since all translated examples belong to class c_l, we have

$$\Pr(C = c_l \mid A = a_2) = 1 = \Pr(C = c_l \mid A = a_2, F_i = 0) = \Pr(C = c_l \mid A = a_2, F_i = 1),$$
$$\Pr(C = \bar{c}_l \mid A = a_2) = 0 = \Pr(C = \bar{c}_l \mid A = a_2, F_i = 0) = \Pr(C = \bar{c}_l \mid A = a_2, F_i = 1).$$

From the above two cases, we can easily conclude that $\Pr(C|\boldsymbol{A}, F_i) = \Pr(C|\boldsymbol{A})$. In other words, F_i and C are conditionally independent given $\boldsymbol{A}$

Clearly, if we can find a set of features $\boldsymbol{A} \subseteq$ Descendant(F_i) satisfying the condition stated in Lemma 4.1, then for any feature F_j that is an ancestor of F_i, F_j and C must also be conditionally independent given $\boldsymbol{A}$, since $\boldsymbol{A}$ is also a subset of Descendants(F_j). Therefore, we give the following corollary, which is derived directly from lemma 4.1.

Corollary 4.1 Let $F_j \in$ Ancestor(F_i) and $\boldsymbol{A} \subseteq$ Descendant(F_i) be a set of features. Let $\boldsymbol{E}$ be the set of translated examples that have value 1's in every feature of $\boldsymbol{A}$. If every translated example in $\boldsymbol{E}$ has the same class label $c_l \in C$, then F_j and C are conditionally independent given $\boldsymbol{A}$.

Based on Lemma 4.1 and Corollary 4.1, we can simply eliminate a conditionally independent feature F_i and all ancestors of F_i if we can find a set of features $\boldsymbol{A} \subseteq$ Descendant(F_i), which satisfies the condition stated in Lemma 4.1. In this situation, F_i and all its ancestors are considered to be subsumed by $\boldsymbol{A}$ as they provide no further information in terms of classification. However, enumerating all feature subsets and conducting a test is still impractical since the number of feature subsets is exponential to the total number of features. To circumvent this problem, we derive Theorem 4.1 that reduces the search space for feature set $\boldsymbol{A}$.

Theorem 4.1 Let $\boldsymbol{B}$ = Child(F_i) and $\boldsymbol{A} \subseteq$ Descendant(F_i). Further, let E_A be the set of translated examples that have value 1's in every feature of $\boldsymbol{A}$ and E_B be the set of translated examples that have value 1's in every feature of $\boldsymbol{B}$. If every translated

example in E_A has the same class label $c_l \in C$, then every translated example in E_B must have the same class label c_l. In other words, if F_i and C are conditionally independent given $\boldsymbol{A}$, then F_i and C must be conditionally independent given $\boldsymbol{B}$.

Proof

Suppose $\boldsymbol{A} \not\subseteq \boldsymbol{B}$ (otherwise this theorem holds). Let $X \in \boldsymbol{A}-\boldsymbol{B}$ and $Y \in \boldsymbol{B}$ be the ancestor set of X. After replacing X by Y, we obtain a new set of features $\boldsymbol{B'} \subseteq \boldsymbol{B}$. Let $\boldsymbol{E_{B'}}$ be the set of translated examples that have value 1's in every feature of $\boldsymbol{B'}$. By downward closure property, $\boldsymbol{E_A} \supseteq \boldsymbol{E_{B'}}$. Therefore, each example in $\boldsymbol{E_{B'}}$ must have the same class c_l. That is, F_i and C are conditionally independent given $\boldsymbol{B'}$. Since $\boldsymbol{B'} \subseteq \boldsymbol{B}$, F_i and C are conditionally independent given $\boldsymbol{B}$.

As a result, for a given feature F_i, we can simply verify its children. If every translated example that has assignment **1** in every child of F_i has the same class, F_i and all ancestors of F_i can be eliminated. Since smaller patterns have more ancestors, verifying features from small to large has the potential of eliminating features in earlier stages. Therefore, features are listed in ascending order of their sizes, and our algorithm sequentially verifies whether a feature and the class variable are conditionally independent. Detailed algorithm (in first stage) is listed below.

FeatureSelection(***T***: a training set; ***F***: a set of features; *N*: integer): ***G***: a set of features
// Suppose features in ***F*** are in ascending order on their sizes
{
//First Stage
    G = ***F***;
    For (each feature F_i in ***G***)
    {
        class0 = 0; class1 = 0;
        For (each distinct translated example *IE* in ***T***)
        {

```
                If IE.Child(F_i) = 1
                {
                    If IE belongs to class 0 {class0++;} else {class1++;}
                }
            }
            // belong to only one member of possible classes
            If ((class0 =0 and class1 ≠0) or (class0≠0 and class1=0))
            {
                G = G – {F_i} – Ancestor(F_i);
            } // end of If statement
        }// end of For loop
//Second Stage
        If |G| > N {G = MarkovBlanketFilter(G);}
        Return G;
}
```

Let us apply *FeatureSelection*() to our example shown in Figure 4.1. Features 1-4 pass our test since they do not have any children. Figure 5-11 pass the test as well because the corresponding patterns of their children are supported by both instances, each having different classes. Feature 12 is the first feature that does not pass the test, because only one instance supports all corresponding patterns of its children– Feature 5, 6, and 8. Therefore, Feature 12 and its ancestor Feature 18 are eliminated. For the same reason, Feature 13, 16, 17, and 19 are eliminated in subsequent steps.

Note that *FeatureSelection()* might require a second stage filtering (*MarkovBlanketFilter()*) because the number of features that pass the first stage (conditional independence test) might still be large. Considering time requirements of computation and the potential redundancy of the remaining features, Markov blanket filter is adopted in this research.

In [KS96], Koller and Sahami formalize their ideas using the notation of a *Markov*

Blanket. We review some of the key concepts here.

Definition 4.5 [Pearl88] Let $\boldsymbol{M}$ be a set of features which does not contain F_i. We say that $\boldsymbol{M}$ is a Markov blanket for F_i if F_i is conditionally independent of $(\boldsymbol{F} \cup C) - \boldsymbol{M} - \{F_i\}$ given $\boldsymbol{M}$

It is easy to see that if $\boldsymbol{M}$ is a Markov blanket of F_i, then it is also the case that the class C is conditionally independent of the feature F_i given $\boldsymbol{M}$. Therefore, if a Markov blanket of F_i can be found, the filter model can safely remove F_i from $\boldsymbol{F}$. Koller and Sahami adopted a greedy strategy for implementing a sequential filtering process in which unnecessary features are eliminated one by one, and it can be shown that a feature tagged as unnecessary based on the existence of a Markov blanket remains unnecessary in later phases when more features are eliminated, as detailed by the Theorem 4.2.

Theorem 4.2 [KS96] Let $\boldsymbol{Y}$ be the current set of features, and assume that some (previously eliminated) feature $F_i \notin \boldsymbol{Y}$ has a Markov blanket within $\boldsymbol{Y}$. Let $F_j \notin \boldsymbol{Y}$ be some feature which is going to be eliminated based on some Markov blanket within $\boldsymbol{Y}$. Then F_i also has a Markov blanket within $\boldsymbol{Y} - \{F_j\}$.

In most cases, however, very few features will have a Markov blanket of limited size. Therefore, constructing an approximate Markov blanket, which is close to the real one, is necessary. The intuition for constructing an approximate Markov blanket is that, if a feature F_i does have a Markov blanket, F_i will directly influence the features of its Markov blanket. Therefore, an approximation, some set of K features which are strongly correlated with F_i, to the Markov blanket can be constructed heuristically.

In [KS96], various "correlation" metrics, including statistical correlation, mutual information, class mutual information, and "pair-wise" cross entropy have been tested. Since our features are binary, statistical correlation is not suitable in this context. Among other three metrics, "pair-wise" cross entropy was shown to have the best experimental results as reported in [KS96]. Thus, we adopt it in our research, shown in Eq 4.3.

$$\Gamma_{i,j} = \sum_{Fi,Fj} \Pr(F_i, F_j) \times D(\Pr(C \mid F_i, F_j), \Pr(C \mid F_j)) \quad \text{.....(Eq 4.3)}$$

In order to figure out how close an approximation is to the real one, Koller and Sahami further define the expected cross entropy as shown in Eq 4.4.

$$\Delta(F_i \mid M_i) = \sum_{f_{m_i}, f_i} \Pr(M_i = f_{m_i}, F_i = f_i) \times D(\Pr(C \mid M_i = f_{m_i}, F_i = f_i), \Pr(C \mid M_i = f_{m_i}))$$

$$\text{.....(Eq 4.4)}$$

Clearly, the lower the $\Delta(F_i \mid M_i)$ value, the closer the approximation is. Thus, the feature F_i which has the lowest $\Delta(F_i \mid M_i)$ value is most likely to have a Markov blanket in the remaining features, and thus should be eliminated first. These approximations result in the algorithm *MarkovBlanketFilter()* listed as below.

```
// G is the total set of features and N is the desired number of features
MarkovBlanketFilter(G: a set of features; N: integer): G' : a set of features
{
    For (each feature Fi in G)
    {
```

```
        For (each feature F_j in G)
        {Compute Γ_ij ;}
    }
    R = G;
    While (|R|>N) do
    {
        For (each feature F_i in R)
        {M_i = the set of K features in R-{F_i} that have the smallest Γ_ij values; }
        For (each feature F_i in R)
        {Compute Δ_R(F_i | M_i) ;}
        F_elimination = F_i for which Δ_R(F_i | M_i) is minimal;
        R = R – { F_i };
    } //end of While loop;
}
```

MarkovBlanketFilter() begins by computing the cross entropy for each pair of features. Then the algorithm constructs an approximated Markov blanket of size K for each feature, and eliminates the feature which has the lowest $\Delta_R(F_i \mid M_i)$ value. This process executes iteratively until the number of remaining features is less than user-specified constant N.

Speaking of computational expense, our algorithm shows promise for scalability. The first stage of the algorithm takes O(nmc) time, where n is the initial number of features, m is the number of translated examples, and c is the maximum number of children of a feature. The complexity result is due to the fact that to eliminate a single feature, we must scan all translated examples and then verify one entry (all values of c children features are **1**) in the conditional probability table. Thus, at most n features are scanned in the case that each of eliminated features has no ancestors.

In the second stage of the algorithm, it requires $O(p^2(m+\lg p))$ operations for computing the pair-wise cross entropy matrix and sorting it, where p is the number of features left after the first stage. The subsequent feature selection process requires $O(rpkm2^k)$ time, where r is the number of features to eliminate, and k is the small, fixed number of conditioning features. This complexity result is due to the fact that to eliminate a single feature, we must iterate through all the remaining features (at most p) and for each one select the k features that have the smallest Γ_{ij} values. Computing $\Delta_R(F_i \mid M_i)$ for each feature F_i requires $O(m2^k)$ time to scan through the m translated examples and compute the entries in the conditional probability tables that we must sum over.

Note that the first stage of our algorithm plays an important role in enhancing the performance of the second stage, in terms of both accuracy and running times. Theoretically [KS96], in the second stage, the larger the conditional set (the larger K), the more likely it is to subsume all the information in the feature, thereby forming a Markov blanket. On the other hand, larger conditioning sets fragment the training set into small chunks, reducing the accuracy of the probability and hence cross-entropy estimates. Therefore, there is a trade-off for setting K.

Because a large extent of redundant information has been eliminated in the first stage, there exists a good possibility that smaller conditional set results in a satisfactory approximation. Smaller conditional set reduces the number of chunks and hence also increases the accuracy of cross-entropy estimates. With smaller conditional set, the running time also dramatically decreases since the computation complexity of the second stage is exponential to K. Therefore, the combined approach is particularly

suitable for our problem – a domain with a huge number of features.

4.4 Performance evaluation

The Bureau of National Health Insurance (BNHI) was founded in 1995 for administering the National Health Insurance Program (NHI) in Taiwan. Through risk pooling, BNHI is responsible for providing the public comprehensive medical care such as health prevention, clinical care, hospitalization, resident care and social rehabilitation. As of June 2002, there were more than 21 million individuals enrolled in NHI with a coverage rate of 96%, and more than 16 thousand medical institutions contracted in the program, which were about 92% of medical institutions nationwide [BNHI].

The medical care expenditure of NHI has experienced a high inflation rate since its inception in 1995 [BNHI]. In 1998, the total expenditure of BNHI was NT $267 billion, a figure that exceeds the revenue NT $262 billion and, compared to 1995, represents a 34% increase in total health care expenditure and a 20% increase in health care expenditure per enrollee. In various reports [BNHI], payment system of NHI– principally[6] "Fee-for-service"– has been noted as the main reason that leads to the rapid increase of health care expenditure.

Insurance programs adopting fee-for-service payment method, as described in Section 2.1, have most serious damage from service providers. Clearly, NHI program fits the

[6] From 1998 to 1999, the BNHI has continued to include laparoscopic surgery, liver transplantation, lung transplantation, and home iron-discharging agent pump in the payment standard and fees were adjusted accordingly. There were additional 50 items applicable to case payment. The Global Budget Payment System was applied on dental service as of July 1st 1998, on Chinese medical service as of January 1st 1999, and on dispensary-level Western medical service as of July 1st 2002.

medical insurance type that our research addressed. With the large number of enrollees and rapid increase of expenditure, NHI is a particular interesting platform for investigating our detection model. Therefore, we consulted domain experts and collected some medical data in NHI for evaluating the effectiveness and efficiency of our detection model.

4.4.1 Data collection and preprocessing

According to reports of BNHI [BNHI], medical claims reported from gynecology departments of various hospitals have a rapid increase of expenditure as well as a high ratio of the number of rejected cases to the total number of claims. For this reason, we decided to focus our attention on medical cases from gynecology departments. By consulting physicians of gynecology departments, we further choose Pelvic Inflammatory Disease (PID) as our major target of detection, since PID is the most common disease in gynecology departments, and the diagnosis as well as treatment methods of PID are representatives of gynecology departments.

The directly reported data of BNHI, however, is not employed in our research for several reasons. First, summarized indices, rather than activity-level logs, are often used in current reports and thus unsuitable to be taken as input of our model. Second, BNHI adopts statistics methods for sampling cases, and infers examined results to the whole population. Thus, only sampled cases are examined. Cases, which are not tagged as fraudulent, might be simply because they are not chosen for examination. Besides, due to the limitations of time and resource, many experts only screen cases by a small set of regulation rules. Certain extent of fraudulent and abusive behavior is not clearly identified.

Therefore, rather than using the reported data of BNHI, we collected data from a regional-level hospital, which is contracted in NHI. We initially gathered 2543 patients' data from the gynecology department of the hospital during July 2001 and June 2002. We prepare two data sets– normal and fraudulent– through the following steps.

(1) Filtering out noisy data: Regarding each patient's treating data as an instance, we removed instances which have missing or noise values. In this step, we removed 77 patients' instances.

(2) Identifying activities: Based on the domain knowledge provided by experts, we identified medical activities in the remaining instances. Some activities, such as examination of blood pressure, were routinely preformed and thus discarded. We finally identified 127 medical activities in this step.

(3) Identifying fraudulent instances: Two gynecologists were involved to identify fraudulent instances. They examined all instances, among which 906 instances were judged by the both gynecologists as fraudulent.

(4) Selecting normal instances: We then randomly selected 906 cases from normal instances, each of which two gynecologists also made the same decision, to form our data set. As a result, a total 1812 instances were used in our experiments.

4.4.2 Induction method

Many induction techniques are available in the literature and even in software packages. We can roughly classify them into three categories according to the formats of derived models, namely decision tree/rule (e.g., C4.5 [Quinlan93], CN2 [PN89], and CBA [LHM98]), discrimination analysis [JW92], and neural network approach [RHW86]. The decision tree/rule approach induces a decision tree (or a set of rules)

that describes the induction model between input features and decision outcomes. The discrimination analysis approach derives linear combination functions of input features under normal distribution and equal dispersion assumptions. The last approach, neural network, produces an appropriate set of weighted links according to a pre-determined network topology that differentiates decision outcomes based on input feature values.

The neural network approach is known for its noise-tolerance and fault-resistance. However, being a holistic approach, the neural network approach suffers from its inability to produce interpretable knowledge [NCL99]. The discrimination analysis is a math-based method, whose assumptions are difficult to satisfy in real situations [NCL99]. Thus, in order to interpret discriminating patterns for medical service management, we focus our attention on decision tree/rule techniques in this research. In some cases, a practitioner could certainly employ another induction technique.

Specifically, we adopt *associative classification* (or classification based on association, abbreviated as CBA) [LHM98] as our induction method. In CBA, a data example is translated as a set of (*feature*, *value*) pairs and a (*class-feature*, *class-value*) pair. Each translated example is treated as a transaction containing a set of items (pairs). Association rules, each of which contains (*class-feature*, *class-value*) pair in the right hand side, are discovered if they have certain user-specified minimum support and confidence. The discovered rules are then selected in terms of precedence order, depicted as below, to develop a classifier.

PRECEDENCE ORDER [LHM98]. Given two rules R_i and R_j, $R_i > R_j$ if

(1) The confidence of R_i is greater than that of R_j, or

(2) Their confidences are the same, but the support of R_i is greater than that of R_j, or

(3) Both the confidences and supports of R_i and R_j are the same, but R_i is generated earlier than R_j.

By the above precedence order, clearly, high-confidence rules, even with low supports, still have high precedence to be selected. The developed classifier thus has the potential of accurately classifying a small set of examples that are covered by the high-confidence but low-support rules. This is particular suitable for our domain, because some features are likely to be good indicators to normal or fraudulent behavior (high confidence) but only supported by few examples (i.e., low support). As a result, CBA is adopted as principle induction technique in this research.

4.4.3 Evaluation criteria

To evaluate the detection model, we consider two measures, *sensitivity* and *specificity*, which are often used in the medical diagnosis and the detection of fraudulent behavior [FW97, Lavrac99]. *Sensitivity* is the ratio of the number of undesired cases returned by a system to the total number of undesired cases. *Specificity* is the ratio of the number of desired cases returned by a system to the total number of desired cases. Clearly, the performance of the system is better if it has both higher sensitivity and higher specificity.

These two measures can be graphically illustrated by Figure 4.2, where A is the set of examples that are actually normal and returned as normal, B is the set of examples that are actually normal while returned as fraudulent, C is the set of examples that are actually fraudulent while returned as normal, and D is the set of examples that are

actually fraudulent and returned as fraudulent. In our case, therefore, *sensitivity* is defined as |D|/|C+D|, and *specificity* is defined as |A|/|A+B|.

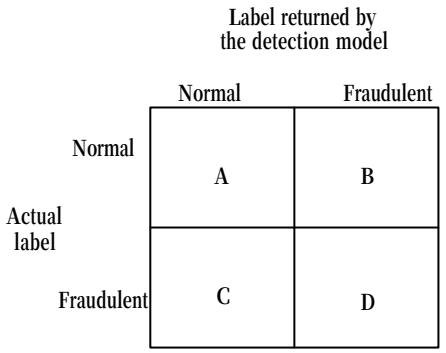


Figure 4.2 A graphical representation

4.4.4 Evaluation results

We used a 5-fold cross validation [JW92] method to evaluate the performance of our detection model. That is, all examples are randomly divided into 5 folds. In each trial, examples in a particular fold are used for testing. Therefore, 5 trials were performed and the overall performance was then estimated by averaging the performance across all trials.

Number of features deducted

In order to construct our detection model, patterns are first discovered by some mining algorithm described in Chapter 3, then translated as features, and finally filtered by the feature subset selection algorithm reported in this chapter. We first report the number of features selected in our model as shown in Figure 4.3. These patterns (features) are discovered at different support thresholds, ranging from 10% to 2% at 2% decrements. Figure 4.3(a) shows the number of initial features (discovered by mining algorithms), and the number of features that pass the first stage of feature

subset selection. Figure 4.3(b) shows the ratio of the number of features that are eliminated by the first stage of feature subset selection to the number of initial features.

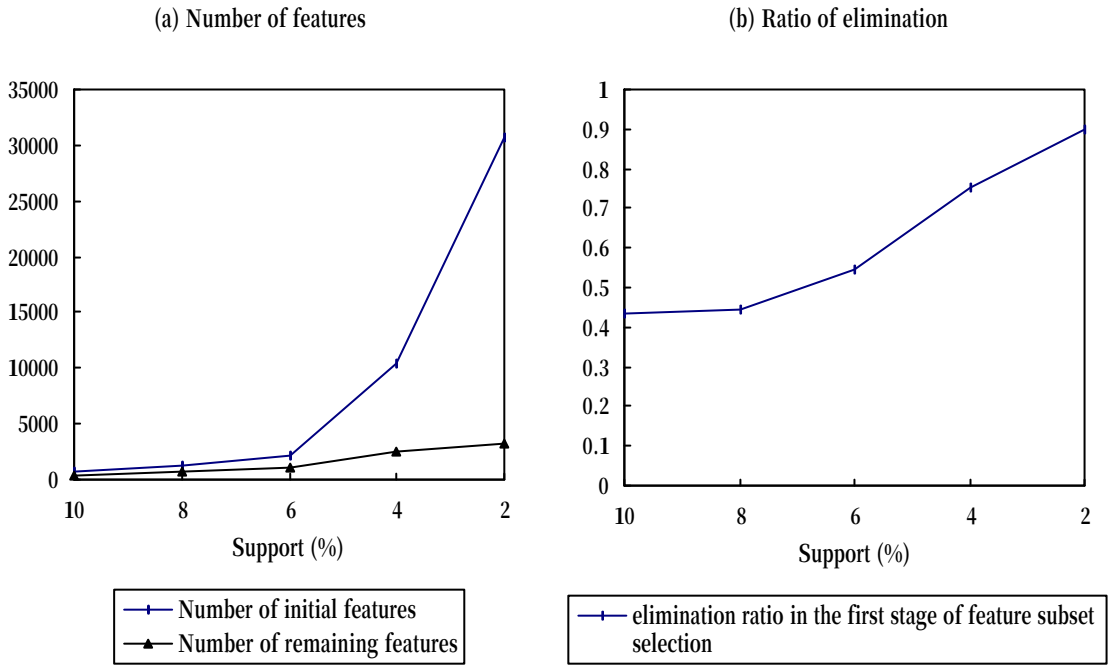


Figure 4.3 Effects of feature subset selection

As expected, the number of initial features increased as the minimum support decreased. While the number of remaining features still increased (at a moderate pace) as a function of support threshold, great percentage of features is eliminated by the first stage of feature subset selection. For example, at 2% support threshold, an average of 30701 features is initially discovered while only 3120 features pass the test. Further, as shown in Figure 4.3(b), the ratio of the number of eliminated features to the number of initial features grows substantially as the minimum support decreases.

Prediction power with the first stage of feature subset selection

We then investigated the sensitivity and specificity of our detection model, which are

constructed by features selected by the first stage of feature selection. At 6% to 2% support thresholds, because the number of features that pass the first stage of feature subset selection is still large (more than 1000), we further filter features by Markov blanket filter (the second stage of feature subset selection) at various levels of Markov blanket size (K=0, 1, 2). 1000 features (N=1000) are finally selected in these cases. Also, since best accuracy of CBA is reported at 1-2% minimum support [LHM98], we set support and confidence of CBA to 1% and 50% respectively. The resultant sensitivity and specificity of our detection model are depicted in Figure 4.4.

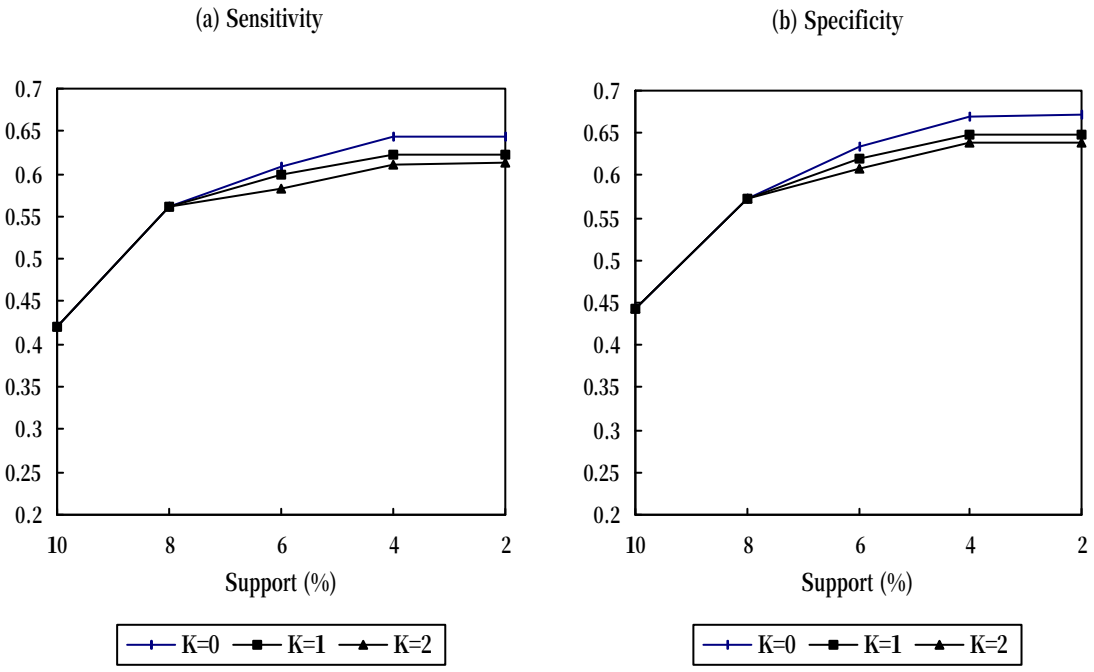


Figure 4.4 Sensitivity and specificity of the detection model with the first stage of feature subset selection

Generally speaking, sensitivity and specificity of the detection model increased as the support threshold decreased. This is expected since higher support threshold indicated

more features were discovered and thus provided more information for classification task. Both best sensitivity 64.36% and specificity 67.12% are obtained at 2% support threshold, while specificity is slightly better than sensitivity. Besides, it is worth noting that both sensitivity and specificity are obtained at K=0 conditioning level, and a slight decrease can be found across various conditioning levels. It shows that a great extent of redundant information has been eliminated in the first stage of features subset selection, and thus a low conditioning level (K=0) is enough to further filter correlated information out.

Prediction power without the first stage of feature subset selection

We also investigated the sensitivity and specificity of our detection model, in which all features were selected by Markov blanket filter with various conditioning settings. The settings of this experiment were the same as the previous one except that the first stage feature selection was omitted. The sensitivity and specificity of the resultant detection model are depicted in Figure 4.5.

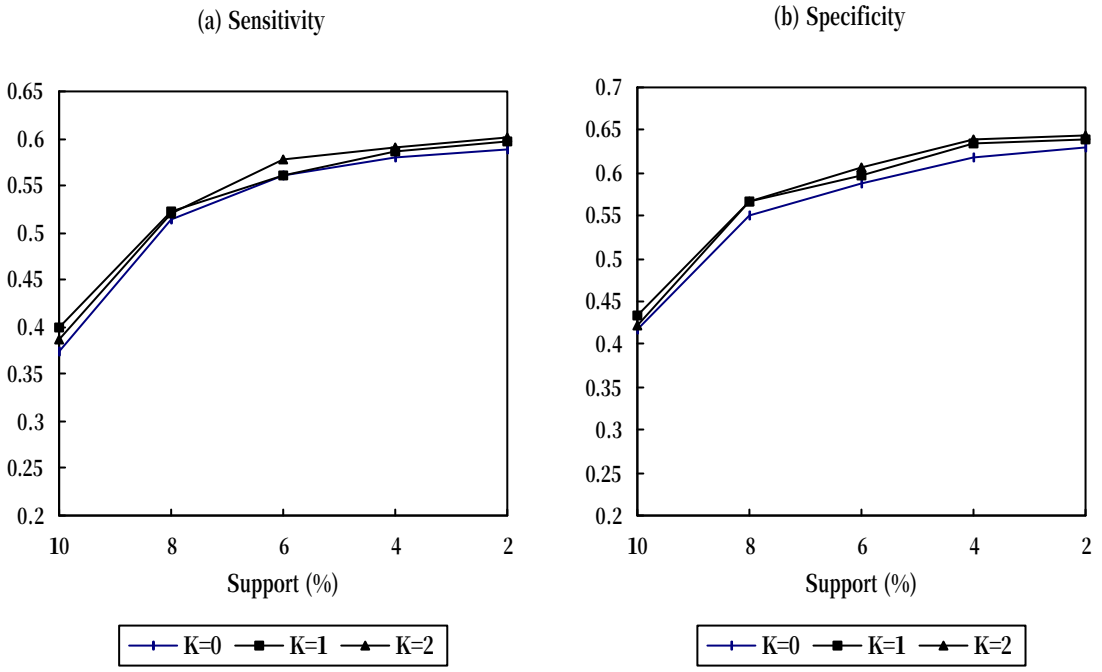


Figure 4.5 The sensitivity and specificity of the detection model without the first stage of feature subset selection

It can be seen that the best sensitivity 60.06% and specificity 64.48% were both obtained at K=2 conditioning level. Clearly, comparing to the experiment shown in Figure 4.4, the performance of the detection model is slightly decreased. It is expected because Markov blanket filter uses only approximations to eliminate features. Besides, the conditioning setting (K=2) shows that it is necessary to have higher conditioning level to filter redundant information, resulting in higher computation time.

Comparison of detection models

We finally compare our detection model with that proposed in [Lan00], which intends to detect suspicious claims in Taiwan's NHI program. We use the same features[7] as identified in [Lan00], which are mainly derived from various expense fields of claims by experts' consultants, to develop an induction model. The resultant sensitivity and

[7] Detailed descriptions are listed in Appendix A.

specificity are shown in Figure 4.6.

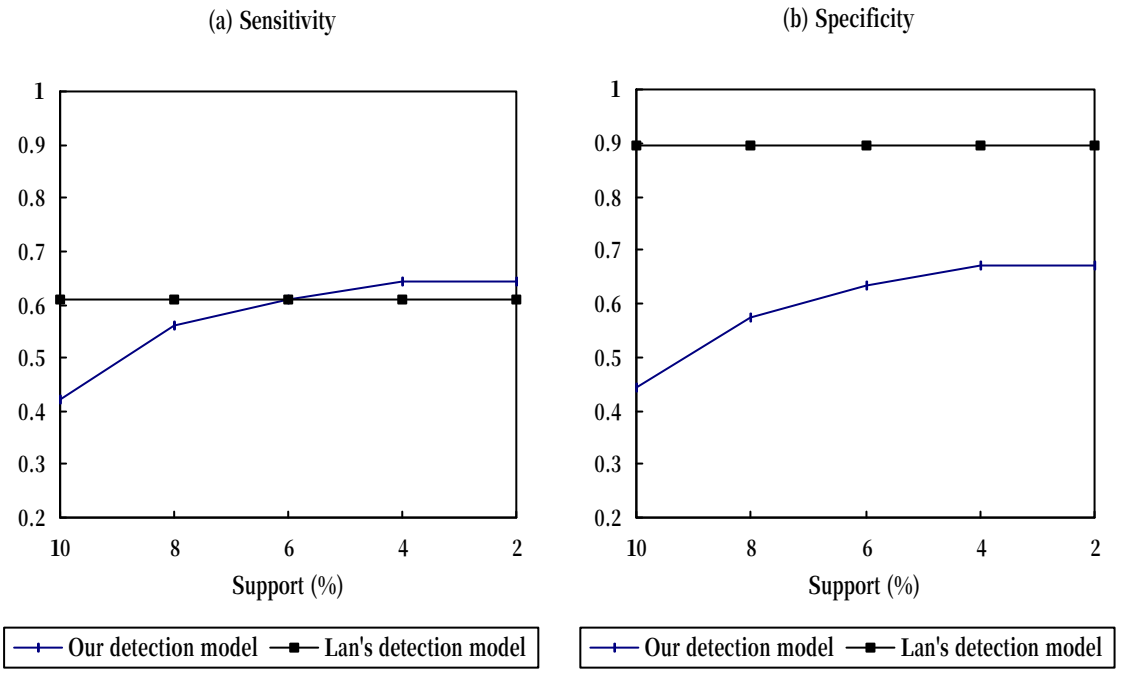


Figure 4.6 Comparisons of detection models

In Figure 4.6, it can be easily seen that Lan's detection model, which mainly involves expense features, has high specificity but low sensitivity. This is because normal examples tend to consistently have low expense, and thus result in high specificity. Fraudulent examples, however, have varied expenses, and thus result in low sensitivity. Similar conclusions are reported in Lan's experiments [Lan00]. Comparing with Lan's detection model, our detection model has a balanced treatment on sensitivity and specificity. Also, the specificity of Lan's detection model is higher than ours, while the sensitivity of our detection model is slightly higher.

The comparison of sensitivity is not intended to show that one is better than the other, but rather to illustrate where the differences lie with our structure feature driven approach. Of fraudulent examples returned by Lan's detection model, our detection

model captures an average of 69% examples. Some examples, such as overdose, are not returned by our detection model. On the contrary, of the fraudulent examples returned by our detection model, Lan's detection model captures an average of 63% examples. Some examples, such as those that have repeated ambulant visits while still have low expense, are not returned by Lan's detection model. Certain extent of differences exists in our structure driven approach and Lan's expense driven approach.

4.5 Summary

In this chapter, we formally defined the feature selection problem, and developed an algorithm for eliminating redundant and irrelevant features. We evaluated the effectiveness of feature selection algorithm and the prediction power of the detection model by using real-world data gathered from Taiwan's NHI program.

The experiments of feature selection showed that great percentage of features could be eliminated by conditional independence test. With the first stage of feature subset selection, the best performance of our detection model is obtained at lower conditioning level, while, without the first stage of feature subset selection, the best performance is obtained at higher conditioning setting. It shows that a great extent of redundant information could be eliminated in the first stage of features subset selection, and result in both accuracy improvement and computation cost reduction.

The experiments of detection models showed that our detection model has certain prediction power. Comparing to Lan's expense driven approach, our structure driven approach performs equally well in both sensitivity and specificity, while Lan's has

better specificity but worse sensitivity. Also, these two approaches tend of capture different fraudulent scenarios. A hybrid approach may have the potential of achieving even better performance but is beyond the scope of our research.

Chapter 5
Model Revision

One key difficulty with induction algorithms is that they require a large, often prohibitive, number of labeled examples to accurately learn the classification model. Labeling clinical instances, which is typically done by domain experts, is a painful, time-consuming process. A traditional detection model that is trained from a small number of labeled examples might thus have less accurate prediction power. Therefore, it is important and interesting to integrate other sources for revising the initial detection model.

Some text classification techniques, such as reported in [NMTM00, WH99, Joachines99], make use of a small number of *labeled* examples and a large number of *unlabeled* examples for creating classifiers with higher accuracy. Since collecting unlabeled examples are much less expensive, we conceive that, by incorporating unlabeled clinical examples into our induction learning process, it may be possible to construct a detection model with higher prediction power.

As a result, in this chapter, we study the problem of integrating unlabeled clinical examples for building a detection model. We first give an overview of related works and identify their theoretical assumptions. We then present a new strategy, which is designed in accordance with the characters of our detection task, to integrate unlabeled clinical examples. The performance of the revised detection model is finally reported.

5.1 Related work

The work of induction learning with labeled and unlabeled data has its origins in statistics and has recently been heavily explored by researchers from machine learning community. This line of research is often conducted through three different approaches: likelihood maximization, discrimination, and co-training approaches.

Most classic methods adopt likelihood maximization approach, which uses an induction algorithm to generate a classifier and Expectation-Maximization (EM) algorithm to estimate class label and parameters of the generated classifier [NMTM00, WH99]. EM algorithm is comprised of two steps: the expectation (E-) step for calculating probabilistically weighted class labels for every unlabeled example, and the maximization (M-) step for estimating new parameters of the classifier. In its implementation form, EM is an iterative process, by which an initial classifier is estimated using the traditional induction algorithm, then the expectation and maximization steps iterate until the generated classifier converges.

As observed in [NMTM00], unlabeled examples might be valuable because they provide information about the generative model of training data even the important class labels are missing. For example, suppose that using only the labeled data a classifier determines that documents containing the word "homework" tend to belong to a positive class. If this fact is used to estimate the classes of unlabeled documents, we might find that the word "lecture" occurs frequently in the unlabeled documents that are now believed to belong to the positive class. This co-occurrence of the words "homework" and "lecture" over the large set of unlabeled data thus provide useful information to construct a more accurate classifier that consider both "homework"

and "lecture" as indicators of positive examples.

The above example imposes an important assumption that the generative model of training data is a *mixture model*[8], which is the key factor for EM to learn with unlabeled data. In text classification, words, which are often used as features in classification, tend to frequently co-occur in documents that belong to the same class. This explains why EM achieves high text classification accuracy [NMTM00] when given only a limited amount of labeled data and a large amount of unlabeled data. On datasets where this assumption is badly violated, however, EM performs poorly [NG00].

As a discrimination approach, Joachines uses transductive support vector machines (Transductive SVMs) to find parameters for a linear separator when given a small amount of labeled data and a large amount of unlabeled data [Joachines99]. The identified linear separator aims to separate the labeled examples of different classes and to maximize the margin over both labeled and unlabeled examples. Joachines also demonstrated the efficacy of this approach for several text classification tasks [Joachines99].

Bennett and Demiriz [BD99] further improved the performance of transductive SVMs when applied to some UCI datasets with computationally easier variant of transduction. It seems that the intuition behind transductive SVMs is that they assume decision boundaries lie between classes in low-density regions of instance space, and that the unlabeled examples help find these areas. However, Zhang and Oles [ZO00]

[8] The term, described in [NMTM00], indicates that features tend to frequently co-occur in the instances that belong to the same class.

argued that both theoretically and experimentally transductive SVMs are unlikely to be helpful for classification in general.

Finally, the co-training setting allows unlabeled data to be used in a new way. It specifies that every example be described by two disjoint views onto the data. For example, with a web classification task, each example has words occurring on a web page, and also words on hyperlinks pointing to web pages. Blum and Mitchell [BM98] present a co-training algorithm that respectively develops two classifiers each based on a different view of data. In the procedure of co-training, the algorithm thus iteratively selects an unlabeled example, gives it a label, and relearns to improve both classifiers. They also show that under the two theoretical assumptions: (1) each set of features is sufficient for perfect classification, and (2) the two feature sets of each example are conditionally independent given the class, an initial (weak) classifier can be arbitrarily improved given sufficient unlabeled examples.

In our problem domain, for evading carriers' detection, service providers tend to use only a few fraudulent or abusive scenarios in one case. Accordingly, it is likely that the assumption of the co-occurrences of fraudulent scenarios (features), a mixture model, is not satisfied. Therefore, we focus our attention on co-training strategy rather than EM algorithm. Certainly, two independent and sufficiently redundant views of data are not available in our detection framework as that in [BM98]. As a result, in this research, we propose a new co-training strategy. Detailed settings and algorithms are discussed in next sections.

5.2 Formalization of model revision problem

In this work, unlabeled clinical examples are integrated into the development of the detection model. Therefore, a set of training examples, only some of them come with class labels while the rest come without class labels, are given. We thus have a disjoint partitioning of the training data, and our view on the whole data set can be formally described as below.

Definition 5.1 We are given a set T of training examples. Only a small subset L of T come with class labels, and the rest U come without class labels. Thus we have a disjoint partitioning of T, such that $T = L \cup U$, and $|L| << |U|$ in general.

According to [Brodley93, Ting94], the success of a developed induction algorithm in finding good generalization for a given data set depends on two factors. The first is whether the algorithm's representation space, such as the Disjunctive Normal Form of a decision tree algorithm, contains a good generalization. The second factor is the goodness of search bias, such as Least Mean Square (LMS) or Absolute Error Correction rule (ACR) in learning the weights of a linear discrimination function [DH73], of an induction algorithm. Due to different representation and search biases, induction algorithms thus have distinct performances [Brodley93].

To achieve better performance, one would expect two induction algorithms to complement each other in that they use different representations and/or search biases for building their classifiers. As such, the two induction algorithms would inform different characters of the data and be able to select (and label) some unlabeled data for the other. The performance of learning task thus could be improved via the

complementation of different induction algorithms as well as the augmentation of labeled data. We capture this intuition and accordingly propose a new co-training strategy, which involves two different induction algorithms.

Let $C_r(-)$ denote a trained classifier. The goal of our model revision problem can thus be stated as below.

Problem statement. Given two different induction algorithms *X* and *Y*, and a set of training examples $T = L \cup U$. The goal of model revision problem is to combine *X* and *Y* to form a classifier $C_r(-)$ on *T*, which has higher accuracy than both $C_r^X(-)$ and $C_r^Y(-)$ on *L*.

5.3 Model revision algorithms

In our co-training strategy, rather than two views of data as those in [BM98], we use two different induction algorithms. The two induction algorithms *X* and *Y* are initially trained on the labeled data *L* to obtain base classifiers. Then using certain criterion, $C_r^X(-)$ selects some of unlabeled data, denoted as L_Y, to label for the other classifier $C_r^Y(-)$. By relearning on all labeled data $L \cup L_Y$, a new classifier $C_r^Y(-)$ trained by induction algorithm *Y* is obtained. The same method is used for *X* to obtain a new $C_r^X(-)$. We repeat this process, and obtain two resulting classifiers. Detailed co-training algorithm is listed as below.

CoTraining($T = L \cup U$: a set of training examples): two classifiers
{
$L_X = L_Y = \boldsymbol{f}$;
$w_X = w_Y = 0$;

Do
$L'_X = L_X$; $L'_Y = L_Y$;
Run induction algorithm X on $L \cup L_X$ to obtain classifier $C_r^X(-)$;
Run induction algorithm Y on $L \cup L_Y$ to obtain classifier $C_r^Y(-)$;
$(L_Y, w_Y) =$ *SelectUnlabeledData*$(C_r^X(-), T, L_Y, w_Y)$;
$(L_X, w_X) =$ *SelectUnlabeledData*$(C_r^Y(-), T, L_X, w_X)$;
While $(L'_X \neq L_X$ or $L'_Y \neq L_Y)$

Return $C_r^X(-)$ and $C_r^Y(-)$;
}

Clearly, the key issue in designing the above co-training algorithm is the criterion for selecting unlabeled data to label (i.e., the procedure *SelectUnlabeledData*()). Since certain extent of prediction errors is likely to be made by both of the two induction algorithms, it is necessary to take additional care. We accordingly discuss this issue, and the detailed algorithm *SelectUnlabeledData*() will be listed in Section 5.3.1. Also, we propose an algorithm *CombineClassifier*(), in which the two classifiers returned by *CoTraining*() are combined, to predict an incoming example in Section 5.3.2. The goal of *CombineClassifier*(), certainly, is to correctly classify the data portions that are correctly classified by either of the two classifiers that are being combined.

5.3.1 Selecting unlabeled examples

Due to mixed data distribution, it is often observed that an induction algorithm has different performances on different portions of data. For example, consider the graphical representation, in which a positive example is marked as a filled circle while

a negative one is marked as an empty circle, of the data set shown in Figure 5.1. If feature F is used to learn the classifier, one induction algorithm might accordingly make its predictions as follows: an example is positive if $F \geq 100$, or equivalently, an example is negative if $F < 100$. Let portion A denotes the set of examples, each of which has value larger than 100 in F, and portion B denotes the set of examples, each of which has value smaller than 100 in F. Using this classifier to predict data, it is easy to see that one would be more confident to the predictions of examples in portion A than to those in portion B.

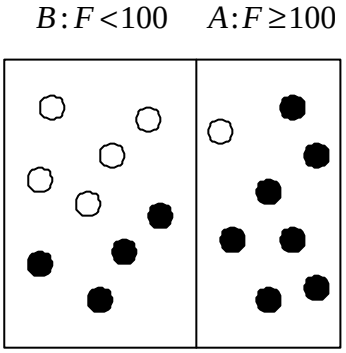


Figure 5.1 Pictorial representation of a data set

This observation suggests an approach of expending labeled data set. Our intuition is as follows: Assume an induction algorithm outputs a classifier that defines a partition of the training data (i.e., CBA partitions the example space into several groups, one defined by each rule), we can think that the induction algorithm may have better performances on some groups. Unlabeled examples that map to the groups with higher accuracy tend to be predicted more accurately. Accordingly, giving the corresponding class labels to these unlabeled examples tend to make less mistakes in mislabeling.

Therefore, in this research, we assume that we have two different induction algorithms, each outputting a classifier that defines a partition of the training data. Let G be a set of groups partitioned by an induction algorithm, we accordingly select a subset G' of the groups that have been evaluated to have higher classification accuracy. We then give class labels for those unlabeled examples that are mapped to G', and put them in the corresponding data sets (i.e., L_X or L_Y). By doing so, we expand the labeled data set for further relearning.

Now, we present in detailed the selection of groups with higher classification accuracy. In its implementation form, we first use a 5-fold cross validation to compute the accuracy of each group. We then define a total order on all groups as below. Certainly, if a group has higher precedence, its predictions are thought to be relatively higher, and hence it should be selected earlier.

Definition 5.2 Given two groups, g_1 and g_2, $g_1 > g_2$ (also called g_1 precedes g_2 or g_1 has higher precedence than g_2) if

(1) The accuracy of g_1 is greater than that of g_2, or

(2) Their accuracies are the same, but the number of labeled examples in g_1 is greater than that of g_2.

Because noise in the labels is likely increase as we extend labeled data set, in addition to precedence order, we design a criterion to control the classification noise rate. This criterion is based on the relationship between the sample size (m), classification noise rate (η), and worst-case accuracy ($1-\varepsilon$) of a classifier, developed in Computational Learning Theory [AL88][KV94] as in Eq. 5.1.

$$\frac{1}{\varepsilon^2} = m \times (1 - 2\eta)^2 \quad \text{.....(Eq. 5.1)}$$

This relationship is used in this research to determine whether the amount of additional data labeled is sufficient to compensate for the increase in noise (mislabeled data). For a classifier *Y*, we can compute the values of *m* (the total number of labeled examples) and η (the classification noise rate), and the resultant equation is listed in Eq. 5.2.

$$Q_Y = |L \cup L_Y| \times (1 - 2(\frac{w_Y}{|L \cup L_Y|}))^2 \quad \text{.....(Eq. 5.2)}$$

where w_Y is the estimated number of mislabeled examples.

When it comes to deciding whether a group *g* should be used to label additional data from *U* for *Y*'s current classifier, we can compute an accuracy estimate Q_g by using Eq. 5.3. If $Q_g > Q_Y$, *Y*'s current classifier will be improved if the examples in *g* are labeled by *X* and added to L_Y.

$$Q_g = |L \cup L_Y \cup U_g| \times (1 - 2(\frac{w_Y + w_g}{|L \cup L_Y \cup U_g|}))^2 \quad \text{.....(Eq. 5.3)}$$

where w_g is the estimated number of mislabeled examples in *g*. Let a_g be the accuracy of *g*, w_g can be computed by $(1 - a_g) \times |U_g|$.

Therefore, for a set of groups partitioned by an induction algorithm, we first sort them in terms of their precedence orders. We then sequentially verify for each group

whether it passes the classification noise test. Only the groups that satisfy the criterion will be used for labeling data for the other classifier. We accordingly design the algorithm of *SelectUnlabeledData*() as below.

SelectUnlabeledData(*Cr*: a classifier, $T=L\cup U$: a set of examples, *L'*: a set of examples, *w*: a real number): *L'*: a set of examples, *w*: a real number

{

Use *Cr* to partition *T* into a set *PU* of groups;

Use *L* for 5-fold cross validation to compute the accuracy a_g for each group *g* in *G*;

Sort the groups in *G* by their precedence orders;

$Q = |L \cup L'| \times (1 - 2\frac{w}{|L \cup L'|})^2$;

For (each group *g* in *G*)

{

$w_g = (1 - a_g) \times |U_g|$;

$Q_g = |L \cup L' \cup U_g| \times (1 - 2\frac{w + w_g}{|L \cup L' \cup U_g|})^2$;

If $Q_g > Q$

{

Let L_g be examples in U_g as labeled by *Cr*;

$L' = L' \cup L_g$;

$w = w + w_g$;

} // end of If statement

} // end of For statment

Return (*L'*, *w*);

}

Take Figure 5.2 as an example. Suppose there are two induction algorithms *X* and *Y*,

each generating a base classifier from 50 labeled examples. Figure 5.2(a) shows *X*'s 40 groups listed in descending order of their precedences, and Figure 5.2(b) shows the mapping of the unlabeled examples to *X*'s groups. Initially, $L_Y = 0$, and we can compute $Q_Y = |50| \times (1 - 2(\frac{0}{50}))^2 = 50$. Therefore, by sequentially testing groups as shown in Figure 5.2 (c), only the first three groups pass the classification noise test, since $Q_{g4} = 49.23 < Q_Y$. As a result, only unlabeled examples that map to groups 1-3 will be labeled and put in L_Y.

Partition groups on L	Accuracy
Group 1	0.8
Group 2	0.7
Group 3	0.65
Group 4	0.6
.	.
.	.
Group 40	0.4

(a) The accuracy of each partition group

Partition groups on U	Unlabeled examples
Group 1	IE 1-20
Group 2	IE 21-40
Group 3	IE 41-60
Group 4	IE 61-80
.	.
.	.
Group 40	IE 145 - 150

(b) The unlabeled examples

L_Y	w_Y	U_{pu}	Q
f	0	f	$Q_Y = \lvert L \cup L_Y \rvert \times (1 - 2(\frac{w_Y}{\lvert L \cup L_Y \rvert}))^2 = 50$
f	0	IE 1-20	$Q_{g1} = \lvert L \cup L_Y \cup U_g \rvert \times (1 - 2(\frac{w_Y + w_g}{\lvert L \cup L_Y \cup U_g \rvert}))^2 = 54.91$
IE 1-20	4	IE 21-40	$Q_{g2} = \lvert L \cup L_Y \cup U_g \rvert \times (1 - 2(\frac{w_Y + w_g}{\lvert L \cup L_Y \cup U_g \rvert}))^2 = 54.44$
IE 1-40	10	IE 41-60	$Q_{g3} = \lvert L \cup L_Y \cup U_g \rvert \times (1 - 2(\frac{w_Y + w_g}{\lvert L \cup L_Y \cup U_g \rvert}))^2 = 52.50$
IE 1-60	17	IE 61-80	$Q_{g4} = \lvert L \cup L_Y \cup U_g \rvert \times (1 - 2(\frac{w_Y + w_g}{\lvert L \cup L_Y \cup U_g \rvert}))^2 = 49.23$

(c) The selection of partition groups

Figure 5.2 An example illustrating *SelectUnlabeledData*()

Further, in our domain, the number of fraudulent and abusive instances in the unlabeled data set is difficult to estimate. It is likely that such incidents are small compared of the amount of "normal" data. Therefore, the work of expanding labeled data set should be able to respond to a skewed class distribution. We accordingly adjust the selection order of partition groups to balance the numbers of fraudulent and normal examples. Specifically, we count two quantities C_n : the number of examples that are now labeled as normal, and C_f : the number of examples that are now labeled as fraudulent. If, in the current stage, C_n is smaller than C_f, groups that are labeled "normal" will have higher priority to be selected, and vice versa. The revision of *SelectUnlabeledData*() is shown as below.

SelectUnlabeledDataR(*Cr*: a classifier, $T=L\cup U$: a set of examples, *L'*: a set of examples, *w*: a real number): *L'*: a set of examples, *w*: a real number
{
Use *Cr* to partition *T* into a set *G* of groups;
Use *L* for 5-fold cross validation to compute the accuracy a_g for each group *g* in *G*;
Sort groups in accordance with their precedence orders;
G_n = {the groups labeled "normal" in *G*};
G_f = {the groups labeled "fraudulent" in *G*};

$$Q = |L \cup L'| \times (1 - 2\frac{w}{|L \cup L'|})^2;$$

$C_f = C_n = 0$;
While ($G_n \neq \boldsymbol{f}$ and $G_f \neq \boldsymbol{f}$)
{
If ($C_f > C_n$)
{*next* = the first group in G_n; $G_n = G_n$ - *next*;}
Else
{*next* = the first group in G_f; $G_f = G_f$ - *next*;}

$$w_{next} = (1 - a_{next}) \times |U_{next}|;$$

$$Q_{next} = |L \cup L' \cup U_{next}| \times (1 - 2\frac{w + w_{next}}{|L \cup L' \cup U_{next}|})^2;$$

```
        If Q_next > Q
        {
            Let L_next be examples in U_next as labeled by Cr;
            L' = L' ∪ L_next ;
            w = w + w_next ;
            If (C_f > C_n ) { C_n = C_n + |L_next − L'|;}
            Else { C_f = C_f + |L_next − L'|;}
        } // end of If statement
    }
    Return (L', w);
}
```

Take Figure 5.2 as an example again. Similarly, we will test partition group 1 and expand the labeled data set with *IE* 1-20 since $Q_{g1} = 54.91 > Q$. Due to the fact that examples in group 1 have "normal" label, group 3, rather than group 2, will be subsequently tested in *SelectUnlabeledDataR*(). If it passes the classification noise test, group 2 will be tested, otherwise group 4 will be tested subsequently. It is worth noticing that the selected data set in *SelectUnlabeledDataR*() might be different with that in *SelectUnlabeledData*(). Certainly, both of them satisfy the criterion, while the former has the advantage of a balanced class distribution, the later has the potential of labeling more unlabeled data.

5.3.2 Combining resulting classifiers

We now describe how the two resulting classifiers of *CoTraining()* are combined. For making a prediction of an incoming example *e*, we compare the accuracies of $C_r^X(-)$, the group of $C_r^X(-)$ that contains *e*, $C_r^Y(-)$, and the group of $C_r^Y(-)$ that contains *e*. The label of *e* is predicted by the maximum of these four quantities. Detailed algorithm of the combination of two resulting classifier $C_r^X(-)$ and $C_r^Y(-)$ is listed

as below.

CombineClassifier($C_r^X(-), C_r^Y(-)$: two classifiers, *e*: an example): prediction result

{

Use *L* for 5-fold cross validation to compute the accuracy a_x of $C_r^X(-)$;

Use *L* for 5-fold cross validation to compute the accuracy a_y of $C_r^Y(-)$;

Use *L* for 5-fold cross validation to compute the accuracy of each partition group defined by $C_r^X(-)$ and $C_r^Y(-)$;

Let *g* be the group defined by $C_r^X(-)$ that contains *e* and a_g be the accuracy of *g*;

Let *g'* be the group defined by $C_r^Y(-)$ that contains *e* and $a_{g'}$ be the accuracy of *g'*;

$M = \text{Max}\{a_x, a_y, a_g, a_{g'}\}$;

Return the label of *M*' s corresponding classifier;

}

5.4 Performance evaluation

In this section, we evaluate the performance of the proposed co-training algorithms for revising the detection model.

5.4.1 Data collection and induction algorithms

We used the data collected from the same regional hospital. We initially gathered 2105 patients' data during October 2000 and June 2001. After removing 55 instances that have some missing data, we finally made an unlabeled data set of 2050 instances.

As a result, we prepare two data sets- a labeled data set of 1812 instances (as described in Chapter4) and an unlabeled data set of 2050 instances- to construct our experiments

For the two induction algorithms, which referred to *X* and *Y* in our general co-training algorithm, we use CBA and C4.5. Clearly, both of these two algorithms satisfy the property that their classifiers form a partition of data, i.e., CBA partitions the data with one group defined by each rule and C4.5 partitions the data with one group characterized by a leaf node. Further, they have different strategies for selecting rules. C4.5 uses the criterion based on information gain to split data set, and adopts a "one feature at a time" greedy strategy for building its classifier. On the contrary, CBA, as described in Chapter 4, finds all rules and then selects the best rules to cover the training cases. Therefore, the rules selected by CBA are "global" best [LHM98]. Due to the fact that they both satisfy the partition property and have different search biases, we explore the co-training effect of CBA and C4.5 in our experiments.

5.4.2 Evaluation results

For each of the following experiments, we start with process structure mining algorithm and feature selection algorithm for constructing the feature set from the labeled data. After translating the labeled data by the feature set, we use CBA and C4.5 to learn the respective initial detection models. Then we explore various strategies to obtain the revised detection model. Therefore, parameters of related algorithms are set as follows: the support threshold of process structure mining is 2%, and conditional independence test and Markov blanket filter (K=0) is used to select 1000 features. The support and confidence of CBA are respectively set as 1% and

50%.

Also, we used 5-fold cross validation for evaluating the accuracy of a detection model. That is, all labeled examples are randomly divided into 5 folds. In each trial, labeled examples in a particular fold are used for testing. Therefore, 5 trials were performed and the overall performance was then estimated by averaging the performance across all trials.

The performance of our co-training algorithm

We first investigate the performance of our co-training algorithm. The performance, by using *SelectUnlabeledData()* and *SelectUnlabeledDataR()* respectively, is reported at different sizes of the labeled data set, ranging from 600 to 1400 at 200 increments. Figure 5.3(a) shows the sensitivities of the initial detection model and the revised detection models by applying *SelectUnlabeledData()* and *SelectUnlabeledDataR()*. Symmetrically, Figure 5.3(b) shows the specificity of the initial and revised detection models.

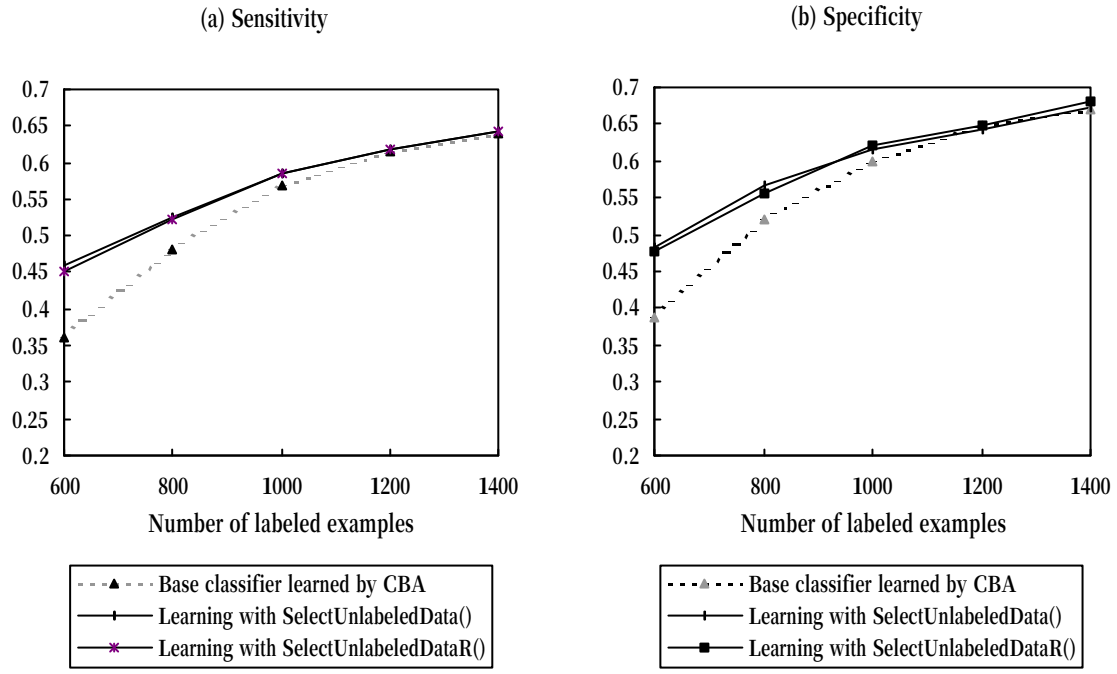


Figure 5.3 The performance of the co-training algorithm

In Figure 5.3, generally, both the sensitivity and specificity of the revised detection model increase with the number of labeled data. With small amounts of labeled data, the co-training algorithm yields more accurate detection model. With a large amount of labeled data, accurate detection model can be obtained without the use of unlabeled data, and the two methods (with/without unlabeled data) begin to converge.

Besides, comparing the two modules of expanding labeled data set (*SelectUnlabeledData()* and *SelectUnlabeledDataR()*), we observe only slightly difference on the sensitivity and specificity at varying amounts of data being labeled. For example, at 600 labeled examples, averaging 490 unlabeled examples are labeled (328 as normal and 162 as fraudulent) by *SelectUnlabeledData()*, while averaging 364 unlabeled examples are labeled (188 as normal and 176 as fraudulent) by *SelectUnlabeledDataR()*. Finally, the co-training algorithm often labels a significant amount of data in each round and thus only requires a small number of iterations. For

example, in our dataset, at most 3 rounds are executed.

Self – training

Also, we are interested in knowing whether the two induction algorithms indeed complement each other. Do the two different induction algorithms use unlabeled data only as well as that one induction algorithm is used? We therefore design this experiment, in which only one induction algorithm is involved. In this experiment, we use CBA as the induction algorithm and *SelectUnlabeledDataR()* as the module of expanding labeled data set. Similar with the co-training algorithm, we initially use CBA to learn a base classifier, and use *SelectUnlabeledDataR()* module to expand the labeled data set. Then, the same induction algorithm CBA is used again to relearn on all labeled data to obtain a new classifier. The process is repeated until the resulting classifier is obtained. The resultant sensitivity and specificity of the revised detection model are depicted in Figure 5.4.

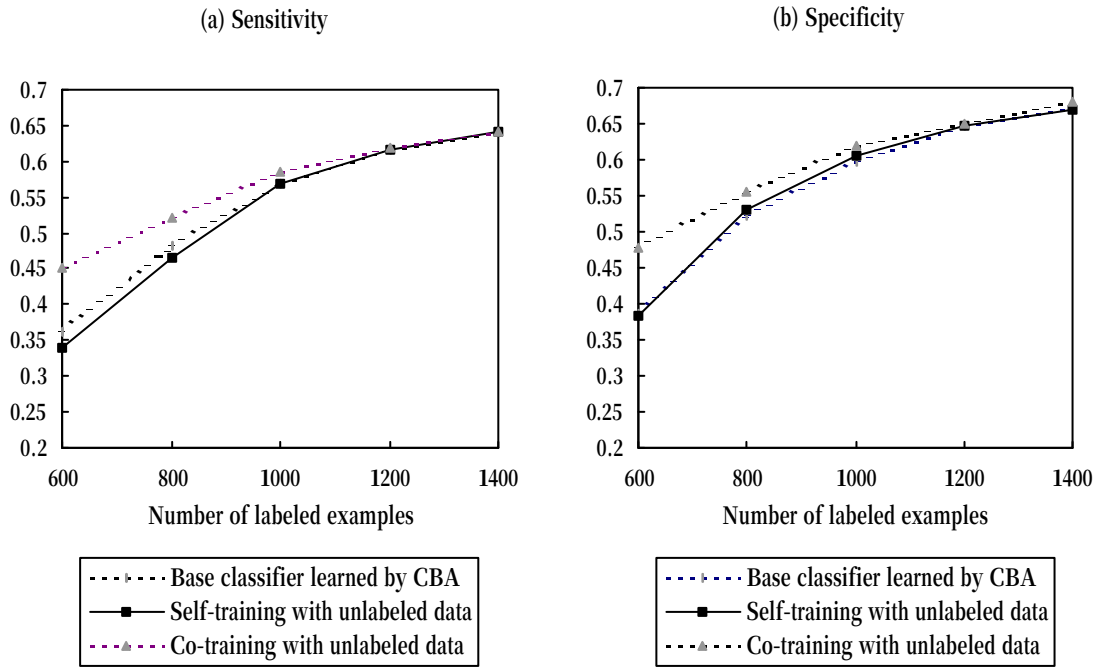


Figure 5.4 The performance of self-training procedure

In Figure 5.4, it can be easily seen that the performance of the self-training procedure is not as good as the proposed co-training algorithm. In some cases, such as those with small amounts of labeled data, the sensitivity of the self-training procedure is even slightly worse than the base classifier (learning with only labeled data). From this experiment, it shows that the complementation of two different induction algorithms plays an important role in the success of our co-training strategy.

Combining classifiers without unlabeled data

We next investigated the effects of the augmentation of labeled data. The question is: Will the combination of two base classifiers (that use only labeled data) perform as well as the co-training algorithm that make use of both labeled and unlabeled data? We therefore design this experiment, in which two settings are compared. In the first setting, we use *SelectUnlabeledDataR()* module to select unlabeled data and two induction algorithms, CBA and C4.5, to co-training the detection model. In the second setting, two base classifiers, which are respectively learned by CBA and C4.5 from only labeled data, are combined to make predictions. The resultant sensitivity and specificity of the revised detection model are depicted in Figure 5.5.

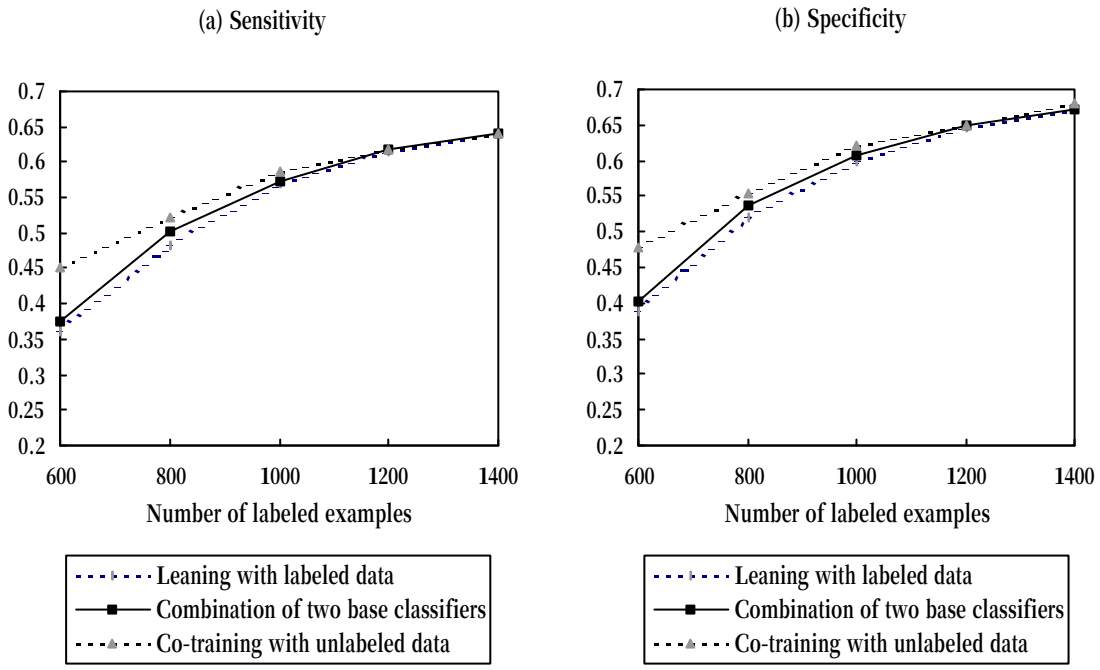


Figure 5.5 The performance of the combination of base classifiers

Clearly, as shown in Figure 5.5, the performance of the combination of two base classifiers is not as good as the proposed co-training algorithm. It shows that the augmentation of labeled data contributes to the improvement of the performance of learning task. Also, the combination of two base classifiers has generally better performance than a single classifier.

Model revised by likelihood maximization approach

We finally apply likelihood maximization approach to our problem. Specifically, we use the famous EM/Naï ve Bayes setting [NMTM00]. In the EM/Naï ve Bayes setting, initial parameter estimates are set using standard Navie Bayes algorithm by just the labeled examples. The EM algorithm, which containing E- step for calculating class labels for every unlabeled example and M- step for estimating new classifier parameters, then executes iteratively until the classifier converges. We use this setting on our data set, and report the resultant sensitivity and specificity, as shown in Figure

5.6.

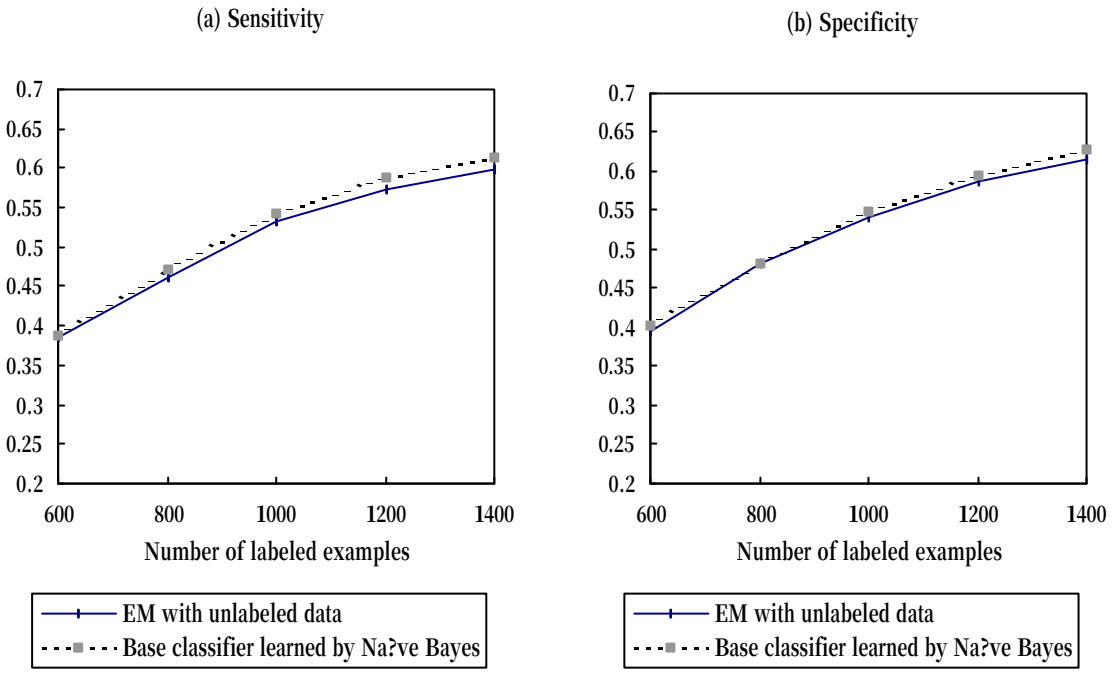


Figure 5.6 Comparisons of the proposed Co-training algorithm and EM algorithm

In Figure 5.6, it can be seen that the base classifier learned by Naï ve Bayes achieves better sensitivity and specificity than that learned with EM procedure at almost every number of labeled examples investigated. In our data set, clearly, EM hurts the performance. This result confirms that reported in [CC02]: EM degrades classification performance if the assumption of mixture model is misfit. Also, comparing with the experiment result depicted in Figure 5.3, the proposed co-training algorithm outperforms the EM/Naï ve Bayes setting.

5.5 Summary

In this chapter, we formally described the model revision problem, and developed a new co-training strategy for improving the prediction power of detection model. We

evaluated the effectiveness of the proposed algorithms by using real-world data, some of which come with labels and rests come without labels.

The experiments of model revision showed that the proposed co-training algorithm improved the performance of the revised detection model, especially with small amounts of labeled data. Besides, comparing with the detection model trained by EM/Naïve Bayes setting, the proposed co-training algorithm yields better performance in our problem domain.

Chapter 6
Conclusion

6.1 Summary

This research described a framework, MCI HCFAD, for Mining Clinical Instances for Health Care Fraud and Abuse Detection. MCI HCFAD consists of process structure pattern discovery, feature selection, and induction programs. Using MCI HCFAD, frequent patterns computed from a set of clinical instances are used to construct predictive features, from which a detection model is inductively learned to detect service providers' fraud and abuse.

We first motivated our research by stating the importance of the detection of service providers' fraud and abuse in the overall health care systems. We provided background on health care fraud and abuse, and briefly described methods of detecting service providers' fraud and abuse in current insurance programs. We pointed out that current methods lack efficiency, adaptability, and extensibility because of the pure knowledge engineering development approaches.

The goal of this research is therefore to develop a framework that facilitates automatic and systematic construction of adaptable and extensible detection systems. For the purposes of building such detection systems, we have studied the problems of mining frequent patterns from clinical instances, selecting features that have more discriminating power and revising detection model to have higher accuracy with less labeled instances. Performance evaluation on the efficiency of the structure pattern discovery algorithms, the accuracy of the detection model in the wake of feature

selection, and the accuracy improvement of the revised detection model are reported.

6.2 Contributions

This research contributes to both the data mining and health care fraud and abuse detection fields.

(1) **Techniques for efficient detection of health service providers' fraud and abuse**. We studied the problem of how to efficiently detect health service providers' fraud and abuse. We propose an induction scheme, which utilizes selected features, to discriminate between normal and suspicious clinical instances. The proposed scheme is efficient, adoptable, and extensible to detect health service providers' fraud and abuse.

(2) **Objective evaluation**. We participated and collected real-data from a health service providers' fraud and abuse detection program. To the health care fraud and abuse detection field, we showed the advantages of automatically and systematically involving knowledge discovery. To the data mining field, we showed the strengths as well as limitations of current techniques and algorithms.

(3) **Techniques for structure pattern discovery**. We studied the problem of how to efficiently discover frequent structure patterns from a large amount of pathway instances. We developed a novel algorithm, and extended two algorithms, Apriori and AprioriAll, whose original goals are to discover association rules and sequential patterns respectively. In addition to pathway instances, the proposed algorithms can be used to discover frequent patterns from interval-based events.

(4) **Techniques for automatic feature selection**. We studied the problem of how to efficiently analyze and eliminate features, which are translated from discovered patterns. We designed a simple algorithm, which is information-theoretic based and guided by probability reasoning, to select optimal feature subset. We have applied this algorithm to construct "fraud and abuse" related features.

(5) **Techniques for induction learning with unlabeled data**. We studied the problem of how to revise the initial detection model with unlabeled data to obtain higher accuracy with less labeled data. We designed a new co-training strategy, in which two different induction algorithms are involved. We also designed a criterion based on Information Theory to expand labeled data set. In addition to revising detection model, the proposed strategy can be also used in learning task under different application domains.

6.3 Limitations

Here, we list limitations, resulted from our research purpose and the proposed methods:

(1) **The focus of our framework is to detect service providers' fraud and abuse**. As discussed in chapter 2, health care fraud and abuse from service providers take the greatest damage. Some types of fraud schemes (i.e., surgeries, invasive testing, and certain drug therapies, etc.) even affect the health of patients. Therefore, we limit our attention in the detection of service providers' fraud and abuse.

(2) **Fee-for-service insurance programs are the application targets of our research**. In cost containment systems, such as case payment or global budget, service providers receive payments (or a budget limit) before giving care services. Underservicing, instead of fraud and abuse, behavior thus occur and become the major concern of insurance carriers. Since our framework is to detect service providers' fraud and abuse, Fee-for-service insurance programs, which is by far the most popular scheme, are our application targets.

(3) **Knowledge engineering task is not involved in the framework**. In order to detect fraud and abuse automatically and systematically, we exploit the power of knowledge discovery rather then knowledge engineering. In some cases, certainly, a practitioner could integrate the knowledge engineering task (e.g., identifying more features by consultants) to detect more accurately. However, this is beyond the scope of this research.

(4) **Only fraudulent and abusive behavior, which exhibits specific structure property, will be detected**. Since only discovered structure patterns are regarded as predictive features, other types of behaviors, which are undesired but not exhibit specific structures, such as billing of overdose, will not be detected.

6.4 Future works

There are several important and interesting future directions:

(1) **Theoretical foundation of the co-training strategy**. Clearly, there exist certain variations to the proposed co-training algorithm that would be interesting to study,

and we believe that further improvements are achievable. In the future, we hope to develop a theory about the co-training procedure so that the capability and applicability of co-training can be understood.

(2) **Cost-sensitive detection model**. There are many cost factors in health care fraud and abuse detection, for example, the cost of computing features and checking a rule (which we have addressed), the cost of identifying an fraud (the labor cost of insurance programs called upon to take actions), and the cost (the damage) of a fraud, etc. These are practical issues that need to be considered when deploying a detection system. The research challenge here is to build detection models that can be easily adjustable according to site-specific "cost policies".

(3) **Integration of detection model with cost-restricted system**. We believe that it is important, beneficial, and natural to integrate a health care fraud and abuse detection system with a cost-restricted system. A lot of fraud and abuse behavior can be limited by a cost-restricted system first because this is part of its function. On the other hand, when detecting fraud and abuse, the detection model can communicate with the cost-restricted system to take appropriate actions, e.g., developing some regulation rules.

APPENDIX A

Discriminating features identified in [Lan00]:

Feature Name
1. Case Type
2. Department Type
3. Patient Type
4. Partial Payment Type
5. Drug Days
6. Physician Gender
7. Drug Fee
8. Diagnosis Fee
9. Examine Fee
10. Drug Administration Fee

LIST OF REFERENCES

[AD94] H. Almuallim, and T. Dietterich, "Learning Boolean concepts in the presence of many irrelevant features," *Artificial Intelligence*, Vol. 69 No. 1-2, 1994.

[AGL98] R. Agrawal, D. Gunopulos, and F. Leymann, "Mining Process Models from Workflow Logs," *Proceedings of the International Conference on Extending Database Technology (EDBT)*, 1998.

[AL88] D. Angluin, and P. Laird, "Learning from noisy examples," *Machine Learning*, Vol. 2, 1988.

[AS94] R. Agrawal, and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proceedings of the International conference on Very Large Data Bases*, 1994.

[AS95] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proceedings of International Conference on Data Engineering*, 1995.

[BD99] K. Bennett and A. Demiriz, "Semi-supervised support vector machines," *Advances in Neural Information Processing Systems*, Vol. 11, 1999.

[BL97] A. Blum, and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, 1997.

[BM98] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," *Proceedings of International Conference on Computational Learning Theory*, 1998.

[BNHI] The Bureau of National Health Insurance (BNHI). Http://www.nhi.org.tw.

[Brodley93] C. E. Brodley, "Addressing the Selective Superiority Problem: Automatic Algorithm/Model Class Selection," *Proceedings of International Conference on Machine Learning*, 1993.

[BWJ98] C. Bettini, X.S. Wang, S. Jajodia, and J.L. Lin, "Discovering Frequent Event Patterns with Multiple Granularities in Time Sequences," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 2, 1998.

[CC02] F. G. Cozman and I. Cohen, "Unlabeled Data Can Degrade Classification Performance of Generative Classifiers," *Proceedings of International Conference on Artificial Intelligence*, 2002.

[CF94] R. Caruana, and D. Freitag, "Greedy attribute selection," *Proceedings of International Conference on Machine Learning*, 1994.

[CH00] D.J. Cook and L.B. Holder, "Graph-based Data Mining," *IEEE Intelligent Systems*, Vol. 15, No. 2, 2000.

[CLR89] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, "Introduction to Algorithms", *MIT Press*, 1989.

[Datta98] A. Datta, "Automating the Discovery of AS-IS Business Process Models: Probabilistic and Algorithmic Approaches," *Information Systems Research*, Vol. 9 No. 3, 1998.

[DH73] R. Duda, and P. hart, "Pattern Clasification and Scene Analysis," *Wiley*, 1973.

[Fukunaga90] K. Fukunaga, "Introduction to Statistical Pattern Recognition," *Academic Press*, 1990.

[FW97] C. P. Friedman and J. C. Wyatt, "Evaluation Methods in Medical Informatics," *Springer-Verlag*, 1997.

[Glaser91] W. Glaser, "Health insurance in practice: international variations in financing, benefits, and problems," *San Francisco: Jossey-Bass Publisher*, 1991.

[Guinane97] C. Guinane, "Clinical care pathways: tools and methods for designing, implementing, and analyzing efficient care practices," *New York:*

McGraw-Hill, 1997.

[HAIPAP98] L. Healy, M. Ayers, R. Iorio, D. Patch, D. Appleby, and B. Pfeifer, "Impact of a Clinical Pathways and Implant Standardization on Total Hip Arthroplasty," *The Journal of Arthroplasty*, Vol. 13 No. 3, 1998.

[Hall96] C. Hall, "Intelligent Data Mining at IBM: New Products and Applications," *Intelligent Software Strategies*, Vol. 7 No. 5, 1996.

[HJU90] K. Hogue, C. Jensen, and K. Urban, "The complete guide to health insurance: how to beat the high cost of being sick," *New York: Avon Books*, 1990.

[HWGH97] H. He, J. Wang, W. Graco, and S. Hawkins, "Application of Neural Networks to Detection of Medical Fraud," *Expert Systems with Applications*, Vol. 13 No. 4, 1997.

[HY02] S. –Y. Hang, and W.-S. Yang, "On the Discovery of Process Models from Their Instances," Decision Support Systems, Vol. 34 No. 1 , 2002.

[Ireson97] C. Ireson, "Critical Pathways: Effectiveness in Achieving Patient Outcomes," The Journal of Nursing Administration, Vol. 27 No. 6, 1997.

[JKP94] G. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," *Proceedings of International Conference on Machine Learning*, 1994

[Joachines99] T. Joachines, "Transductive Inference for Text Classification using Support Vector Machines," *Proceedings of International Conference on Machine Learning*, 1999.

[JW92] R. Johnson, and D. Wichern, "Applied Multivariate Statistical Analysis," *Englewood Cliffs: Prentice-Hall*, 1992.

[KL51] S. Kullback, and R. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, Vol. 22, 1951.

[KR92] K. Kira and L. Rendell, "The feature selection problem: Traditional methods and a new algorithm," *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 1992.

[KS96] D. Koller and M. Sahami, "Toward Optimal Feature Selection," *Proceedings of International Conference on Machine Learning*, 1996.

[KV94] M. Keans and U. Vazarini, "An introduction to computational learning theory," *MIT Press*, 1994.

[Lan00] C. H. Lan, "A Data Mining Technique Combining Fuzzy Sets Theory and Bayesian Classifier- An Application of Auditing the Health Insurance Fee for the National Health Insurance," a thesis in *Yuan-Ze University*, 2000.

[Lavrac99] N. Lavrac, "Selected techniques for data mining in medicine," *Artificial Intelligence in Medicine*, Vol. 16, 1999.

[LHM98] B. Liu, W. Hsu, and Y. Ma, "Integrating Classification and Association Rule Mining," *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 1998.

[LLM97] M. Lassey, W. Lassey, and M. Jinks, "Health care systems around the world: characteristics, issues, reforms," *Upper Saddle River: Prentice Hall*, 1997.

[LS94] P. Langley, and S. Sage, "Induction of selective Bayesian classifiers," *Proceedings of the AAAI Symposium on Relevance*, 1994.

[NELH] National Electronic Library for Health. Http://www.nelh.shef.ac.uk

[NG00] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability in co-training, " *Proceedings of International Conference on Information and Knowledge Management*, 2000.

[NHCAA91] "Guidelines to Health Care Fraud," *REPORT, National Health Care Anti-Fraud Association (NHCAA)*, 1991.

[NHCAA02] "Health Care Fraud: A Serious and Costly Reality for All Americans," *REPORT all_about_hcf, National Health Care Anti-Fraud Association (NHCAA)*, 2002.
Http://www.nhcaa.org

[NMTM00] K. Nigam, A. Mccalum, S. Thrun, and T. Mitchell, "Text Classification from Labeled and Unlabeled Documents using EM," *Machine Learning*, Vol. 34, 2000.

[Pearl88] J. Pearl, "Probabilistic Reasoning in Intelligent Systems," *San Mateo: Morgan Kaufmann*, 1988.

[PN89] P. Clark, and T. Niblett, "The CN2 Induction Algorithm", *Machine Learning Journal*, Vol. 3 No. 4, 1989.

[Quinlan93] J. Quinlan, "C4.5: Programs for Machine Learning," *Los Altos: Morgan Kaufmann*, 1993.

[RHW86] D. Rumelhart, G. Hinton, and R. Williams, "Learning Internal Representations by Error Propagation, Parallel Distributed Processing: Explorations in the Microstructures of Cognition," MIT Press, 1986.

[SA96] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," *Proceedings of the 5th International Conference on Extending Database Technology (EDBT)*, 1996.

[SCL99] T. Sung, N. Chang and G. Lee, "Dynamics of Modeling in Data Mining: Interpretive Approach to Bankruptcy Prediction," *Journal of Management Information Systems*, Vol. 16 No. 1, 1999.

[SGWRJ01] L. Sokol, B. Garcia, M. West, J. Rodriguez, and K. Johnson, "Precursory Steps to Mining HCFA Health Care Claims," *Proceedings of the Hawaii International Conference on System Sciences*, 2001.

[Sokol98] L. Sokol, "Using data mining to support health care fraud detection," *Proceedings of the International Conference on the Practical*

Application of Knowledge Discovery and Data Mining (PADD), 1998.

[Ting94] K. M. Ting, "The problem of small disjuncts: its remedy in decision trees," *Proceedings of Canadian Conference on Artificial Intelligence*, 1994.

[WA96] V. William and B. Archer, "Medicare program: changes to the hospital inpatient prospective payment systems and fiscal year rates," *REPORT RIN: 0938-AH34, The United States General Accounting Office*, 1996. Http://www.gao.gov

[WH99] Y. Wu and T. S. Huang, "Using unlabeled data in supervised learning by discriminate-EM algorithm," *Proceedings of the Workshop on Using Unlabeled Data for Supervised Learning*, 1999.

[ZO00] T. Zhang and F. J. Oles, "A probability analysis on the value of unlabeled data for classification problem," *Proceedings of International Conference on Machine Learning*, 2000.

LIST OF PUBLICATIONS

Journal papers

1. S. –Y. Hwang, W. –C. Hsiung and W. –S. Yang, "A Prototype WWW Literature Recommendation System for Digital Libraries," to appeared in Online Information Review (SSCI), 2003.

2. S.-Y. Hwang, C.-P. Wei and W.-S. Yang, "Discovery of Temporal Patterns from Process Instances," to appeared in *Computer in Industry* (SCI expanded), 2003.

3. S.-Y. Hwang and W.-S. Yang, "On the Discovery of Process Models from their Instances," *Decision Support Systems* (SCI expanded), Volume 34, Issue1, December 2002, Pages 41-57.

Conference papers

1. W.-S. Yang, 2002, "Process Analyzer and Its Application on Medical Care," *Proceedings of 23nd International Conference on Information Systems (ICIS02) Doctoral consortium*, Spain.

2. W.-S. Yang, 2002, "Process Pattern Discovery and Its Application on Clinical Pathway Analysis," *Proceedings of 6th Pacific Asia Conference on Information Systems (PACIS02) Doctoral Consortium*, Japan.

3. S.-Y. Hwang, C.-Y. Hong and W.-S. Yang, 2001, "Mining Generalized Profile Association Rules in Support of New Product Recommendations," *Proceedings of 10' th International Workshop on Information Technologies and Systems (WITS01)*, New Orleans, Louisiana.

4. S.-Y. Hwang, J.-K. Chiu and W.-S. Yang, 2001, "Personal Workflow Management in Support of Pervasive Computing," *Proceedings of the 2' nd International Conference on Mobile Data Management*, Hong Kong.

5. C.-P. Wei, S.-Y. Hwang and W.-S. Yang, 2000, "Mining Frequent Temporal Patterns in Process Databases," *Proceedings of 10' th International Workshop on Information Technologies and Systems (WITS00)*, Australia.

6. W.-S. Yang and S.-Y. Hwang, 1999, “Mining Instance Data to Discover Process Model,” *Proceedings of 9’th International Workshop on Information Technologies and Systems (WITS99)*, Charlotte, North Carolina.