AN AUTOMATED SYSTEM FOR GENERATING SITUATION-SPECIFIC DECISION SUPPORT IN CLINICAL ORDER ENTRY FROM LOCAL EMPIRICAL DATA

Jeffrey G. Klann

Submitted to the faculty of the University Graduate School
in partial fulfillment of the requirements
for the degree
Doctor of Philosophy
in the School of Informatics
Indiana University

September 2011

Accepted by the Faculty of Indiana University, in partial
fulfillment of the requirements for the degree of Doctor of Philosophy.

Gunther Schadow, MD, PhD, Chair

Stephen M. Downs, MD, MS

Doctoral Committee

John T. Finnell, MD, MS

June 22, 2011

Matthew J. Palakal, PhD

Peter Szolovits, PhD

DEDICATION

To those who have helped me regain my life - may I give something back.

ACKNOWLEDGEMENTS

This PhD has been a trailblazing experience. Being among the first students in this particular PhD program has given me support and flexibility to design a unique research direction. Regenstrief Institute selected me as their first pre-doctoral fellow, which has not only been an honor but also has put me footsteps away from many of the greatest clinical informatics researchers in the nation. Peter Szolovits at MIT has graciously agreed to continue mentoring me from a distance, adding his particular genius to the mix.

I would also like to thank the other members of my committee. A particular thanks goes to Gunther Schadow and Stephen Downs. Professor Schadow has been involved with all my research since the beginning of the program, and coming to understand his outside-the-box approach to difficult problems has been essential to my success in research. Professor Downs has been a mentor even before I decided to come to Indiana, and his deep knowledge of Bayesian networks has been invaluable to my research. Professor Finnell has been a great support to me and his involvement has on several occasions calmed my fears that this would never come together. His enthusiasm for the success of his students is a great comfort to me. Professor Palakal has supported me since my first days in the program, and provided a great deal of academic advising though he was not my academic adviser. He walked through the PhD process with me.

Many others inside the IUPUI community contributed to this work as well. I would like to thank: JM McCoy, who served on my committee until he left the University a year ago and helped me formulate my ideas early in my PhD, when my excitement about 'wisdom of the crowd' was just a fledgling idea; Marc Overhage, for his leadership and insights which inspired several of the broad strokes to this work; Jeff Warvel, for making the Gopher data available and for spending innumerable hours explaining its intricacies to me; Joe Kesterson, for providing results and demographics data; Siu Hiu, for helping me devise reasonable statistical methods to evaluate my data; and Jon Duke, for providing immeasurable assistance in understanding clinical medicine.

I am privileged to have had contributions by several people outside of the Indiana University community as well. H.Y. Baksh-Mohammed, nurse practitioner, contributed information on common treatments for complications of pregnancy. Greg Cooper at the

University of Pittsburgh helped me mature my ideas regarding fading in learning (used in TF-TCDBNs). Harold Lehmann at Johns Hopkins introduced me to Condorcet's jury theorem and voting theory. Clark Glymour at Carnegie Mellon University helped me to understand several details of Bayesian structure learning. Joseph Ramsey, also at Carnegie Mellon University, answered many questions regarding Tetrad and its source code. Abhik Shah, PhD student at University of Michigan, gave me insight regarding the functionality of his Pebl tool. John Stutz at NASA helped me to understand the limitations of BNT (the Bayes Net Toolbox). I must thank Daphne Koller and Nir Friedman's excellent book, Probabilistic Graphical Models, as well. Even though it is specifically cited only occasionally, its comprehensive overview of Bayesian network approaches gave me much of the depth of my knowledge on the subject.

It is difficult for words to express my gratitude toward my wife and her support of this project. Not only did she come with me to Indiana for this, and not only did she take over most of the house chores as I've been slaving away at LaTeX, but she also always had a hug for me when I felt everything was going to fall apart. Her hugs were essential to the completion of this project.

Thank you to the rest of my family as well, especially to my parents. They believed in me and supported me throughout this program, from when I first thought about returning to school through my virtual disappearance for the first half of 2011 while I finished my dissertation.

Finally, I would like to thank the One who made everything not made by human hand, for giving us such an intricate world to explore and in which to create. His genius is beyond comprehension and his mentorship has changed my life. I can only hope that some tiny sliver of His mind is reflected in what I do.

This research was supported in part by grant 5T15 LM007117 from the National Library of Medicine.

ABSTRACT

Jeffrey G. Klann

AN AUTOMATED SYSTEM FOR GENERATING SITUATION-SPECIFIC DECISION SUPPORT IN CLINICAL ORDER ENTRY FROM LOCAL EMPIRICAL DATA

Clinical Decision Support is one of the only aspects of health information technology that has demonstrated decreased costs and increased quality in healthcare delivery, yet it is extremely expensive and time-consuming to create, maintain, and localize. Consequently, a majority of health care systems do not utilize it, and even when it is available it is frequently incorrect. Therefore it is important to look beyond traditional guideline-based decision support to more readily available resources in order to bring this technology into widespread use. This study proposes that the wisdom of physicians within a practice is a rich, untapped knowledge source that can be harnessed for this purpose. I hypothesize and demonstrate that this wisdom is reflected by order entry data well enough to partially reconstruct the knowledge behind treatment decisions. Automated reconstruction of such knowledge is used to produce dynamic, situation-specific treatment suggestions, in a similar vein to Amazon.com shopping recommendations. This approach is appealing because: it is local (so it reflects local standards); it fits into workflow more readily than the traditional local-wisdom approach (viz. the curbside consult); and, it is free (the data are already being captured).

This work develops several new machine-learning algorithms and novel applications of existing algorithms, focusing on an approach called Bayesian network structure learning. I develop: an approach to produce dynamic, rank-ordered situation-specific treatment menus from treatment data; statistical machinery to evaluate their accuracy using retrospective simulation; a novel algorithm which is an order of magnitude faster than existing algorithms; a principled approach to choosing smaller, more optimal, domain-specific subsystems; and a new method to discover temporal relationships in the data. The result is a comprehensive approach for extracting knowledge from order-entry data to produce situation-specific treatment menus, which is applied to order-entry data at Wishard Hospital in Indianapolis.

Retrospective simulations find that, in a large variety of clinical situations, a short menu will contain the clinicians' desired next actions. A prospective survey additionally finds that such menus aid physicians in writing order sets (in completeness and speed). This study demonstrates that clinical knowledge can be successfully extracted from treatment data for decision support.

Gunther Schadow, MD, PhD, Chair

TABLE OF CONTENTS

CURRICULUM VITAE

LIST OF TABLES

LIST OF FIGURES

LIST OF PROGRAMS

NOMENCLATURE

2-TBN	Two-slice Temporal Bayesian Network
AUC	Area Under the Curve (usually refers to area under the ROC)
BK	Boyen and Koller's DBN Inference Algorithm
BN	Bayesian Network
BNT	Kevin Murphy's Bayes Net Toolbox
BS	Blanket Stitch (Bayesian Network Structure Learning Algorithm)
CBN	Causal Bayesian Network
CDA	Clinical Document Architecture
CDS	Computer Decision Support
CDSS	Computer Decision Support System
CHF	Congestive Heart Failure
CPOE	Computer Physician Order Entry
CPT	Conditional Probability Table
CS	Collider Sets (Bayesian Network Structure Learning Algorithm)
CTBN	Continuous-Time Bayesian Network
DBN	Dynamic Bayesian Network
EBM	Evidence-based medicine
ED	Emergency Department
EM	Expectation Maximization Parameter Estimation
GES	Greedy Equivalence Search (Bayesian Network Structure Learning Algorithm)
GLIF	Guideline Interchange Format
GS	Grow-Shrink (Markov Blanket Discovery Algorithm)
HIT	Health Information Technology
IAMB	Iterative-Associative Markov Blanket (Markov Blanket Discovery Algorithm)
ICU	Intensive Care Unit
INPC	Indiana Network For Patient Care
JUNG	Java Universal Network/Graph Framework
MAG	Maximal Association Graph

MICU	Medical Intensive Care Unit
ML	Maximum Likelihood Parameter Estimation
MMHC	Max-Min Hill Climbing (Bayesian Network Structure Learning Algorithm)
PC	A Constraint-based Bayesian Network Structure Learning Algorithm
PPV	Positive Predictive Value
QMR	Quick Medical Reference
RMRS	Regenstrief Medical Record System
ROC	Receiver-Operator Characteristic (plot of true positive rate vs. false positive rate)
SQL	Structured Query Language
TF-TCDBN	Time-forward Temporal-Causal Dynamic Bayesian Network
UVC	Urgent Visit Clinic
W	Wilcoxon sum-of-ranks statistic
WVC	Women's Visit Clinic

PART 1

Treatment Advice from the Crowd

CHAPTER 1

Introduction

1. Problem Statement

Despite the consistent steady increase of health information technology (HIT) systems in hospitals (Ford et al., 2006), health care costs continue to rise (Medicare & Medicaid Services, 2006) and currently-implemented HIT systems offer no clear benefit (Himmelstein et al., 2010) or path to return-on-investment (Ash et al., 2004). However, one component of HIT, computer decision-support (CDS), has repeatedly been shown to improve the quality of health care delivery (Kaushal et al., 2003) and reduce costs (Kaushal et al., 2006). CDS could be a key to making HIT investments worthwhile. Unfortunately, CDS usage lags in existing HIT systems. A 2005 Massachusetts-wide survey showed that less than 50% of healthcare institutions have and utilize CDS even some of the time (Zhou et al., 2009; Simon et al., 2007), with only a 3% increase by 2007 (Simon et al., 2009). A major reason for this is that CDS content is, with great time and expense, manually created (Waitman, 2004), maintained (Geissbuhler and Miller, 1999), and localized (Garg et al., 2005). Not infrequently, even when CDS is available, the content is inappropriate (Van der Sijs et al., 2006) or does not account for complex clinical situations (Sittig et al., 2008). To combat these roadblocks, it is important to look beyond traditional guideline knowledge to other sources of decision support.

In particular, CDS systems have overlooked the local wisdom of physicians within a practice as a valid knowledge source. Long before HIT, physicians had a tradition of informally sharing expertise, knowledge, and local practice standards through collegial conversations in lunchrooms and hallways (Perley, 2006). Studies have shown that physicians value colleagues' advice nearly as much as textbooks and sometimes even prefer it (Haug, 1997). Dr. Cathy Perley suggests the main reason for this is that medicine is locally situated, and colleagues can provide a local frame of reference through which to decide if and how external (global) guidelines relate to particular (local) cases (Perley, 2006). Perley hypothesizes that the overwhelming amount of medical knowledge is often distilled into usable and applicable chunks through these informal conversations (also referred to colloquially as 'curbside consults').

Because computer physician order entry (CPOE) systems are gaining widespread use, we believe that much of this collegial knowledge is being implicitly captured in order records. We believe these order records are rich enough to partially reconstruct the knowledge behind treatment decisions. If this is true, this local knowledge resource could be utilized for situational, dynamic treatment recommendations based on the data of the local practice. This 'electronic curbside consult' is similar to shopping recommendations provided by web sites like Amazon.com. Such recommendations are not vetted by professional review, but are produced statistically by correlating recent purchases to items that similar customers have also purchased (Linden et al., 2003). On Amazon.com, the result is a personalized shopping experience, increased sales, and improved customer loyalty (Schafer et al., 2001). Applying this approach to medicine has similar revolutionary potential. For one, because the knowledge is local, Perley suggests it might be more palatable to clinicians than external guidelines. Secondly, this knowledge is *already* being captured and is already localized and automatically updated, reducing cost and time dramatically over traditional decision support. Thirdly, it might be able to capture recommendations for complex comorbidities that guidelines have trouble addressing and may be a major detractor to existing CDS (Sittig et al., 2008; Van der Sijs et al., 2006).

Such recommendations would have several possible direct uses. One, it would provide physicians a new decision support resource, allowing a quick informal check of their treatment plan against those of similar cases in their community without interrupting workflow to perform a physical curbside consult. Two, such checks could be performed automatically, generating alerts when an action falls far outside the local standard. Three, in systems which already have access to mature DSS content, it could provide a localization filter to reduce workload in adapting guidelines to the local practice. This use will be important to study once collaborative, coded decision support becomes available through projects like the Decision Support Consortium (Middleton, 2009).

We therefore propose to discover whether situation-specific local treatment recommendations can be automatically, algorithmically generated from the community wisdom in our local order entry data. We will develop a method to translate complex CPOE data into abstracted decision-making models for inpatient medicine. This methodology will utilize

CPOE data as its input (e.g., primary diagnosis, secondary problems, previous orders). Then, we will develop an application that applies this model to produce a customized, ranked list of treatment recommendations. Finally, we will evaluate the resulting lists both through simulation on a test set of actual hospitalizations and through comparison to expert opinion.

2. *Wisdom* of the Crowd?

The idea that the crowd is wiser than the individual is at least as old as Greek democracy, and it continues to proliferate today. As mentioned, Internet users frequently trust the 'wisdom of the crowd' for an impressive variety of tasks. We believe the crowd can provide comprehensive answers to complex questions (e.g., through collaborative encyclopedias like Wikipedia); we look to the crowd to recommend books and movies (e.g., Amazon.com and Netflix); and we understand that the crowd will lead us to the most relevant information (e.g., through page ranks on search engines like Google).

Consumer healthcare is beginning to embrace a crowd-wisdom approach as well. For example, PatientsLikeMe is a website which solicits the ill to provide health information in order to find new cures and treatments. Frost et al. (2011) found that research data provided by consumers on the site can capture trends not seen in randomized controlled trials. For example, in a group of 1755 multiple-sclerosis patients using modafinil, 99% of patients used it for an off-label purpose. Some of these off-label purposes were unexpected or unusual (e.g., 'brain fog'). Similarly, CureTogether[1] offers patients an opportunity to collaboratively find new treatments for their diseases. For example, at the time of this writing, users ranked 'spending time with animals' almost as effective as Xanax in the treatment of anxiety.

However, physicians are, for the most part, reticent to embrace such approaches in mainline healthcare. Perhaps the only notable manifestation of true 'crowd wisdom' in mainline healthcare is the physician discussion board. Most large-scale knowledge-sharing has instead focused on more efficiently propagating expert-curated guidelines to the local practice (e.g., Guidelines.gov or Middleton's (2009) CDS Consortium). As discussed previously, however, there is great need for better-maintained CDS content, and the wisdom

[1] www.curetogether.org

of the local practice is increasingly being captured implicitly in order data. Therefore, it is important to review the theory behind crowd wisdom and understand where it should and should not be used.

There is a widespread assumption that physicians do not naturally 'do the right thing', and so relying on average practice patterns is dangerous. In fact, there is important truth in this assumption. McGlynn et al. (2003) famously showed that physicians provided only 54.9% of recommended care, when judged against 439 quality indicators. Even in encounter-oriented quality measures (where physicians do well), physicians nationally provide recommended care only 73.4% of the time. Physicians admit that time-pressure and stress can cause them to behave inconsistently with their training (Ely et al., 1995).

However, 'crowd wisdom' does not require that each individual in a crowd make the correct decision. Indeed, this is antithetical to the concept. Rather, 'crowd wisdom' is the belief that the group can be wiser than the individual. Condorcet (1785) derived the *jury theorem*, upon which all voting theory is grounded. It proves that when each member in a group of independent decision makers is more than 50% likely to make the correct decision, then aggregating those decisions ultimately leads to the right answer. The number of decision-makers required is frequently very small, often as small as three (King and Cowlishaw, 2007). If we believe that a physician is more likely than chance to make the correct decision, we can trust their aggregated knowledge. (In fact all functions of health care delivery - screening, diagnosis, treatment, follow-up - are performed properly by physicians more than 50% of the time, according to McGlynn et al. (2003).)

A small caveat is that decisions must generally be binary for the original jury theorem to hold. Thankfully, many of McGlynn's quality measures are formulated as binary decisions (e.g., initiate antiplatelet therapy after noncardiac stroke). Moreoever, Arrow (1950) proved that aggregating multimodal decisions frequently still leads to wisdom if *irrelevant alternatives are not independent*. Frequently, in medicine, many irrelevant alternatives are not independent. Medical treatments often happen in groups (e.g., diagnostic radiology or blood test panels), so whole chunks of medical treatments can be eliminated together. Therefore, aggregating knowledge often results in good group decisions even in multimodal situations.

The jury theorem makes intuitive sense in thinking about suggesting medical treatments through aggregated decisions. If physicians make the majority of errors due to tiredness, distraction, and interruption (as the literature might lead one to believe), then reminders regarding average behavior will be correct and allow tired physicians to catch their errors. However, one very important caution remains.

World history has demonstrated 'crowd madness' as well as 'crowd wisdom'. (For example, Hitler was an elected official.) The problem is that, for the jury theorem to hold, decision-makers must be 'sincere' (Austen-Smith and Banks, 1996), meaning essentially that they must be independent decision-makers. It is unlikely that in a large practice each practitioner is making independent decisions. They are influenced by colleagues, formularies, available equipment, and the like. This explains one type of 'group madness' in medicine: 'local practice variation.' Fisher et al. (2009) utilized data from Medicare beneficiaries to show that the quality of care in a region is profoundly influenced by the 'ecology' of healthcare in that region (including resources and capacity, social norms, and the payment environment). This study, part of the Dartmouth Atlas project, found a twofold difference in healthcare volume (the per capita number of tests, prescriptions, etc.) between some neighboring regions. Disturbingly, the study also showed no statistical correlation between increased volume and increased healthcare quality. Figure 1.1 shows this variation in volume by region.

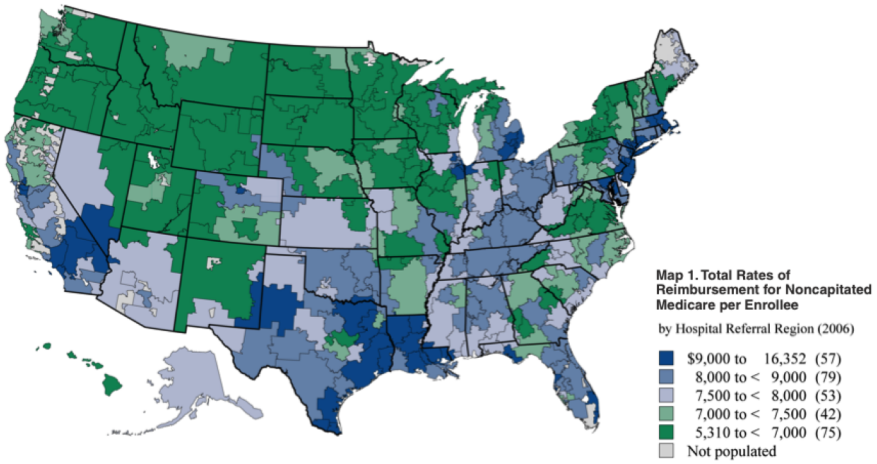


Figure 1.1. Medicare reimbursement by hospital-referral region. From Fisher et al. (2009). Used with permission.

This demonstrates the important role that evidence-based medicine (EBM) must play in CDS: only manually-curated reminders can introduce entirely new knowledge or enforce new behavior. However, I propose this principle: *Where there are not strongly negative external influences, applying majority voting methods to local standards can result in decision-support content at least on par with guideline-based knowledge.* Although crowd wisdom must certainly not replace expert validation or manually-curated reminders, harnessing local data is needed to help maintain CDS to a point where it is actually useful in a larger number of clinical systems.

3. Statement of Contribution

Thus we proceed with caution into harnessing this crowd wisdom for decision support. This dissertation develops and evaluates methods to 'reverse-engineer' CPOE data into CDS knowledge. The evaluation focuses on how well the local practice is reconstructed, not on how well this compares with national guidelines. This first task will occupy us for many chapters, and we leave comparison with national guidelines to future work.

Processing order entry-data to reconstruct local wisdom and generate decision-support content has not been comprehensively explored in existing literature. As we will see in the next chapter, there is growing interest in the *idea* of the secondary use of medical data for decision support, but it has only been lightly explored. Most work has focused on finding

pairs of associated variables in the data. A small amount of work has focused on search or on process discovery.

In this work, I present a new approach to solving this problem by adapting advanced methods that have been untried for this purpose. In particular, I will leverage the power of the probabilistic graphical model, or Bayesian network, in order to create a multivariate model of medical processes that can be reasoned over through probabilistic relations. It will have the following properties:

(1) Multivariate. It will capture relationships that involve multiple interacting variables.

(2) Probabilistic. By associating the multivariate relationships with probabilities, it will support reasoning under uncertainty (e.g., when a clinical situation is not fully known).

(3) Temporal and associative. The model will capture both associations (non-temporal relationships) and processes (temporal relationships) in order to provide a clear picture of treatment suggestions.

(4) Fast. By decomposing a large database into simpler models, this system will be able to reason about the clinical situation in real time.

Previous work all provide at least one of these characteristics, but none offer them all.

Reasoning with Bayesian networks is well-understood, and such networks are already widely used in medicine. However, methods to learn the networks from data are not well explored in this domain, and they have never to our knowledge been used to create decision-support tools from crowd wisdom. This work incorporates several existing learning algorithms in novel ways, and also introduces several new approaches.

4. Data Source and Preparation

The experiments in this dissertation will use data from Wishard Hospital, the county hospital in Indianapolis. Indianapolis overall is one of the most average regions of the country (in terms of healthcare spending and population diversity), so we feel it is a good testbed for the principle of crowd wisdom.

The Gopher Clinical Workstation is a CPOE system that has been in operation at the hospital since 1984 and has expanded to several outpatient clinics. It is ubiquitous in the hospital and clinics, captures a great deal of high-quality data, and has long been used as a testbed for decision support functionality (McDonald, 2002). It captures data at an extremely fine level of granularity, capturing all problems (diagnoses and complaints) and orders entered during a hospitalization grouped temporally by CPOE session (in which orders are placed at one time by a healthcare provider). It also includes patient context such as location within the hospital. Among other useful analyses, the data allow one to look at diagnoses as they are entered by providers during the hospitalization, and not those generated by medical coders long after discharge. The quality and depth of these data make it ideal for a methodological study of CPOE data.

The principal limitation of relying only on CPOE data is that this is not a complete view of the patient. Other important factors, such as the results of previous orders (test outcomes, physiologic changes, etc.) contribute to the clinical context as well. We will explore augmenting our data sources in Chapter 9, but for the most part we find that what a CPOE dataset lacks in breadth it makes up for in completeness, and so it is still a good testbed for local-wisdom-extraction methodologies.

Until recently, the only source for Gopher data was through the Regenstrief Medical Record System (RMRS), which synthesizes data from many different sources (MacDonald et al., 1994). The RMRS stores all of the orders from for a hospitalization, but only the billing diagnoses and none of the temporal or contextual information. A recent project within the Regenstrief Institute has now extracted the raw Gopher data, going back to 2007.

The Gopher development team provided us these data for a three-year period, 2007-2009, in the form of tab-delimited files. We wrote a script in the Python language to eliminate all free text and internal codes (e.g., workstation identifiers). Remaining identifiers, according to 45 CFR A 164.514b, included patient identifiers and dates. We processed these as follows:

(1) Patient identifiers. Patient identifiers were replaced with a random number (a pseudo-id). This pseudo-id was remembered throughout the deidentification process so that the pseudo-id remained consistent for a patient (for correlation across hospitalizations).

(2) Dates. Dates were de-identified according to the following algorithm: Each patient pseudo-id was assigned a random date offset of $\pm 60 - 120$ days. All dates associated with that patient were adjusted by this offset. This algorithm allows relative comparison within a patient (e.g. the average frequency of rehospitalizations), but not actual date comparison across patients.

Our Python script then loaded these data into a PostgreSQL database as two tables: sessions and orders. A session corresponded to an order session within a hospitalization and included a patient identifier, an admission number, a date and time, a location, and in some cases the name of the Gopher form that was being filled out to place orders (e.g. 'admitting orders'). The table included 2,362,152 sessions in 424,667 admissions. The orders table included, for each order placed, the Gopher order number, the action taken ('new', 'continue', and 'discontinue'), a link to the order session, and an field that provides some context about the order's meaning (e.g., is this a chief complaint or a secondary diagnosis?). This table contained 7,677,432 orders. Gopher order numbers connect to the Gopher terminology dictionary, which is a comprehensive catalog of 7376 orders and problems, for which is specified both their broad class (problem, test, radiology, drug, consult, nursing, diet) and more specific subtype and components (if the order is part of a set or panel).

Analysis of the data revealed that each hospital admission was actually a visit to one of three locations: the emergency department (ED), the inpatient hospital, and the women's visit clinic (WVC). Within the ED, less complex patients were triaged to an area called the urgent visit clinic (UVC). We will call these the four *modalities* of medicine in Wishard. We therefore created an admissions table which included a unique admission identifier, a session id within that admission, a link to each order session in the admission, and the modality of this admission. We then wrote a PostgreSQL stored procedure in PL/Python to populate this table using the following rules: 1) A hospitalization has the modality

associated with its initial location; 2) Inpatient hospitalizations also include any sessions that occur immediately before, up to and including an admission note (because the majority of inpatient admissions in Wishard start in the ED); 3) Order sessions in the operating room are excluded (but not the recovery room or the waiting area), because the operating room is a very different environment not particularly compatible with the four general modalities; 4) discharge orders do not always occur at the end of hospitalization, so these are reordered to the end; 5) if no admission id is associated with the session, use the date to determine which hospitalization it occurred in. The fourth point is due to a bug in Gopher - the date of the order session is the date the session was begun, and physicians at Wishard frequently open a discharge session immediately after admission so they will not forget to submit it when discharge occurs. The fifth point addressed 29000 sessions with no admission id tagged with them. Our admission table contained 67800 inpatient, 230971 ED, 36771 WVC, and 63445 UVC visits.

In order to reduce noise in the data, we reduced the orders table in the following ways. Initially the table contained 7376 order types. First, we excluded orders not labeled as problems (diagnoses and complaints are called 'problems' in Gopher and are placed as orders), diets, tests, treatments, procedures, referrals, or consultations. This removed 221 order types, including mostly order types used for testing, very rare orders, improperly entered orders, and call orders. (Call orders, which are physician notification events ordered by the clinical staff, are extremely frequent and we felt their presence would reduce the ability to detect more meaningful patterns). Next we excluded order types that had been deactivated in Gopher, which removed an additional 29 order types. After careful examination, we eliminated an additional 45 vague order types that provided minimal information, all of which ended with the words 'other' or 'misc' or began with the words 'cannot rule out' (e.g., *nursing orders other*). Next, orders placed in the test ward (where training occurs) were eliminated. Next, if individual orders were made when an entire panel containing that order were made in the same session, the individual order was eliminated. This removed only 7 order types but 45,520 actual orders. Finally, we attempted to identify all medications that included the route of administration, and we combined these into a single order

type without the route. We chose our route terms manually, because there was no comprehensive knowledgebase available regarding what orders were synonymous. Therefore this was not perfect, but it did remove 170 order types. A final pass was run to remove rare orders of a few specific types, removing 167 additional order types. In the end, our updated order table contained 6701 order types and 5,787,422 orders. It is possible to reduce this further using order sets defined in Gopher (which group categories of orders), but these are problematic for several reasons: orders which are actually of different classes (e.g., morphine and ibuprofen) are sometimes grouped together; the order sets are largely unmaintained; and, the order sets do not distinguish between panels (orders which occur in a group) and synonyms (orders that are actually equivalent). In Chapter 3, we make some use of these sets, but for the most part we chose not to rely on them.

Next, we created a summary table used for our experiments. The fields in this table are shown in Table 1.1. We will refer to subsets of this table by modality through this work: GopherInpatient, GopherED, GopherUVC, and GopherWVC. Finally we created an admission-compressed version of this table (where sid is not included) to study whole hospitalizations. This dataset was approved by the IRB (#EX0811-29).

Field	Description	Source table
gid	A global encounter id	admissions
sid	A session number within the encounter	admissions
o	An order number from the Gopher terminology dictionary	orders
status	Whether the order is active or being discontinued	orders
modality	The modality associated with this hospitalization	admissions
location	The specific location within the hospital for this session	encounters

Table 1.1. The fields in our Gopher summary table, used for all experiments in the subsequent chapters.

5. Outline of this work

This dissertation is organized into three parts. In Part 1, we will review existing work and develop a first version of a system to select orders in dynamic clinical context. In Part 2, we will make three methodological contributions to the algorithms used. In Part 3, we will synthesize all the pieces into a final system, which we will evaluate both retrospectively and prospectively.

Part 1 includes Chapters 1-4. In Chapter 2, we will review existing techniques in medical data mining and reasoning. In Chapter 3 we will introduce and evaluate a preliminary approach to generating situation-specific treatments using Bayesian network learning. Chapter 4 introduces a more comprehensive approach and more robust evaluation measures. Part 2 includes Chapters 5-8. These develop novel approaches to scalability and temporal reasoning in a Bayesian network framework. Part 3 includes Chapters 9-11, which involve the creation of a final system and a final evaluation of our work.

CHAPTER 2

Data Mining and Reasoning in Medical Informatics

In this chapter, we review two parallel threads of work involving computation in clinical medicine. This includes clinical data mining and clinical reasoning.

1. Clinical Data Mining: A Brief Review

US Healthcare costs have increased about 5% per capita per year over the last four decades, which is double the growth of inflation during that time (Orszag, 2008). In response to this, the federal government has been pushing for increased health information technology (HIT) for the past two decades.

In 1991, the Institute of Medicine released a seminal booklet explaining how HIT in general, and computer-based patient records in particular, are essential for the future of health care (Institute of Medicine, 1991). The booklet praised the potential quality improvement and efficiency savings through HIT. Embracing the promise of this potential, President Bush presented a mandate that healthcare practices switch to fully electronic systems by 2014 (Ford et al., 2006). Although more realistic estimates suggest 86% adpotion by 2024 (Ford et al., 2006), recently the Obama administration introduced Meaningful Use (Blumenthal and Tavenner, 2010), which incentivizes providers to use HIT through increased Medicare reimbursements. Slowly the country is moving to electronic capture and sharing of clinical data.

However, even with the electronic capture of data, the promises of quality and efficiency improvement have been slow to occur. In 2007, a survey from the Medical Records Institute confirmed that HIT was not meeting its goals. 30% of respondents indicated that *no* HIT goals were being met in their environment (Conn, 2007).

There are many technological and non-technological reasons for this sluggish adoption, but one major barrier is methodological. The healthcare community needs data-mining methods to interact with data repositories which can address the particular complexities and pitfalls of healthcare. For this reason, the National Library of Medicine has been funding a great deal of research into medical data collection, sharing, and data mining (Lindberg and Humphreys, 1995).

Still, Bellazzi and Zupan (2008) write that the application of data mining in medicine, "despite high hopes, has until recently been relatively limited." They suggest that the goal of predictive data-mining is various forms of decision support, but our literature review supports that even two years after their article, successes have been mild.

Perhaps for this reason, medical data mining is best known in more specialized tasks, such as data completeness, search, algorithms for specialized use (e.g., ventilator protocols), and non-clinical tools such as epidemic detection (Maciejewski et al., 2008) and genomic association studies. Still, as techniques have improved, there has been a resurgence of interest in using medical data for more general decision support mining. This work has predominantly used algorithms from retail and e-commerce (which we will refer to collectively as *Collaborative Filtering*), though more recent work has utilized process-discovery approaches.

tid	Book	Bookmark	Coffee	Magazine	Tea	Cards
1	Y	Y	Y			
2				Y	Y	
3	Y	Y			Y	
4	Y	Y		Y		
5	Y	Y	Y	Y		
6	Y			Y		
7						Y
8	Y	Y			Y	
9	Y				Y	

Table 2.1. A short transaction set showing purchases among nine customers at a bookstore. Nearly all of the relevant data-mining algorithms operate by performing statistics on tables of transactions like these. In this example, each row represents a transaction (noted by the number in the TID column), and each column represents an item that may have been purchased in that transaction.

Medical data mining has for the most part dabbled in many methods developed in other fields, without engendering development of its own. As previously stated, although there is growing interest in finding associations in medical data, studies which utilize clinical data to provide decision support are relatively rare outside of specialized studies. We will examine methods that have been used for this purpose, turning occasionally to Table 2.1 for reference.

1.1. Data mining as search. Although the research reviewed here focuses on harnessing the collective wisdom of the crowd (e.g. aggregating information) to suggest interventions, one notable alternative use for medical datasets deserves review: search. We will refer to this search-based approach under the umbrella term 'case-based reasoning', in which the current situation is compared with exemplar cases, the most similar of which are displayed. This approach has found specialized applications in medicine (Bichindaritz and Marling, 2006), but it presents two problems.

First, developing exemplar cases is historically done through manually constructed 'gold standard' cases (Althoff et al., 1998), which proffers a similar burden as manually developed CDS. Abidi and Manickam (2002) described a method to extract a case-base directly from an electronic record, but the method first requires manual mapping and is not intended to be automated. Paterson et al. (2005) suggest that the Clinical Document Architecture (CDA) could automate the extraction, but their work focuses on translating from text into CDA format, not decision support with CDA.

Other work has bypassed the problem of finding exemplar cases by utilizing an entire medical database for exemplars, which is a more familiar approach in the modern world of Internet search engines. However, then the second problem of choosing similar cases becomes critical, or many irrelevant results will be retrieved. Popescu and Arthur (2006) describe a system which would allow a physician to search for patients with similar diagnoses, in order to ascertain how they were treated, but to our knowledge it was never implemented or evaluated. Both Melton et al. (2006) and Cao et al. (2008) developed methodologies to find similar patients, and the latter study had greater than 60% correlation to experts.

The search-based approach is certainly needed, but much more work is needed to increase specificity and develop optimized algorithms. Even with perfect specificity, searching for common situations would produce an overwhelming number of results. This can be seen in our bookstore example, where a search for tea-purchasers would, upon user examination, reveal that none of them also purchase coffee, but it would require manually examining four cases. In a real bookstore transaction log, this process could take hours. However, for the rarer event of purchasing cards, only one exemplar would appear. Search-based approaches are an alternative, not a replacement, for automated decision-support content development.

They do not capture the aggregate wisdom of the crowd, but only the wisdom of individuals in specific situations, which is best used in rare situations.

1.2. Collaborative Filtering. In order to capture the wisdom of the entire crowd from data, one needs pattern recognition algorithms, which comprise the remainder of the methods we will discuss.

The most popular technique is so-called collaborative filtering. This term was coined by Goldberg et al. (1992), who used it to refer to users cooperatively sharing e-mail filters. The term now refers to any aggregation of statistical data across transactions to provide feedback to a decision-maker. It is synonymous with the *recommendation algorithm*, which is used most popularly in e-commerce to suggest items to customers by combining their purchase history and preferences with those of other customers (Schafer et al., 2001). Such algorithms take many forms, but the best known might be Amazon.com, which personalizes the entire shopping experience for each user (Linden et al., 2003).

It is no surprise, then, that these methodologies are frequently chosen as a starting point for medical recommendations. They are popular, simple, and fast (for example, they can be used in speed-critical environments like Amazon.com).

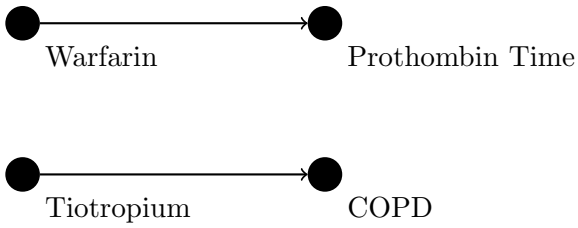


Figure 2.1. An example of two pairwise rules. The first is a corollary order, the second is a medication that might indicate a missing problem.

1.2.1. *Pairwise association.* By far the fastest of all these approaches is to examine only pairs of variables that occur in these transactions. For example, Amazon's algorithm records the number of times some purchase B occurred in the same transaction as A, for all A and B. The number of occurrences is known as the *support* for that association. Dividing the support by the number of occurrences of the individual item A is known as the *confidence*. If the association's confidence exceeds some threshold, then it is used as a recommendation.

For example, Table 2.1 shows that when a book was purchased, a bookmark was also purchased 5 out of 7 times, which we would write as confidence(book → bookmark) = $\frac{5}{7}$. Many heuristics exist to determine the strength of an association (e.g. Lallich et al. 2007), and these are known as *interestingness measures*.

Several studies have utilized pairwise association in clinical data to support decision-making. Chen et al. (2008) has proposed an automated system to mine biomedical literature's MeSH headings for CDSS content, but its evaluation only compared the method across different data sources, not the adequacy of the method. A perusal of the results suggest that her method works well in only very common situations. Carter et al. (2002) successfully used pairwise association in clinical records to *initialize* `may-treat` linkages in RxNorm, but these were only a starting point for content developers.

Two studies have used an experimental design to examine the quality of pairwise associations. The first was our own (Klann et al., 2009), in which we measured perceived clinical utility of the 92 top corollary orders generated by a very fast pairwise association rule mining algorithm. We found that, although 70 of these rules were clinically meaningful, only 44 of these were directly relevant and could potentially be useful reminders.

The most comprehensive experimental study is perhaps the one performed by Wright et al. (2010). They used pairwise association to discover association between drugs and diseases, in order to find problems missing from the problem list that are indicated by the patient's medication profile. They found that the 50 associations with the highest χ^2 value were clinically valid associations. However, many of these were related to HIV/AIDS and all dealt with very specific situations. The clinical accuracy of the associations dropped off very rapidly at lower χ^2 values.

This highlights the weakness of pairwise algorithms: they cannot detect associations that depend on multiple variables. In our example above, for example, books and coffee are frequently purchased together, as are tea and coffee, but books are never purchased with both coffee and tea, and this cannot be detected by pairwise association. In the same way, one might imagine that drug X is highly predictive of problem A unless drug Y is also ordered, which would indicate that drug X is actually treating problem B. To detect situations like this, we need multivariate associations.

1.2.2. *Multivariate association.* Multivariate association techniques find large groups (or itemsets) of co-occurrences within transactions through a process called candidate generation. The first efficient candidate generation algorithm, Apriori, was developed by Agrawal et al. (1994). It performs a breadth-first search of increasingly large item sets in a database of N elements, which comprise the powerset of N, denoted $\mathcal{P}(N)$. Therefore a naive search would comprise 2^N itemsets, which is intractable for large N.

The key insight of Apriori is the downward closure property, which states that any subset of a frequent itemset must be frequent. This allows the algorithm to conclude *a priori* that some paths down the tree are not worth exploring, because they cannot contain frequent itemsets. This allows significant pruning of the search tree at every level, especially in sparsely connected data sets.

Much work has improved upon Apriori, including depth-first searches like Eclat (which often find maximum itemsets more efficiently) and searches within maximal cliques (which can allow for parallelization) (Zaki, 2000). However, frequent itemset searches are difficult to perform in real-time due to their computational intensiveness. Therefore in the commercial world they are typically used for offline data-mining tasks, such as understanding customer purchasing patterns (Schafer et al., 2001).

Multivariate association has shown some promise in the complex environment of medicine. Che (2007) suggested an itemset-mining methodology used to generate order set drafts, though the resulting order sets poorly correlated with existing order sets. Santangelo et al. (2007) explored itemset mining to discover novel laboratory test batteries, but their limited study did not evaluate the clinical utility of the discovered itemsets, except to note that the most common test battery was not included in their results. Wright and Sittig (2006) performed an exploratory study that demonstrated frequent itemset mining on hospital data might be useful in providing order suggestions. Their system found several useful order sets, but they manually culled through many useless associations.

Overall, this methodology has suffered from a lack of robust exploration within medical data, possibly because it is both far more complicated than pairwise association but also not specific enough to provide significant additional value. In particular, these algorithms produce a large number of item sets, and it can be difficult to discern what clinical process

is actually occurring. A more useful approach for treatment recommendations would be to try to extract the clinical process from the data.

1.3. Mining clinical processes. Several studies have done exactly this. Decision-tree learners generate a 'flowchart' of choices (e.g., treatments) lead to classification of a decision (e.g., finding or diagnosis). A typical method is Quinlan's (1993) C4.5 and C5.0 algorithms, which iteratively split a set of transactions into optimal subsets (using a measure called *information entropy*, creating a binary tree of decisions. These flowcharts leading to a classification (e.g., finding or diagnosis). Although the algorithm merely classifies, several studies explore whether decision trees might approximate the actual process used in decision-making. However, even if this is true in some situations, Quinlan notes other limitations in this approach. In particular, the algorithms can only be used to discover *logical classification models*, in which actions are taken based on logical statements about individual variables. He notes that they do not have the flexibility (multivariate interaction, for example) of network models (such as neural networks), which we will turn to shortly. However, they are also much less computationally intensive.

Two studies have used Quinlan's algorithms to discover clinical processes from databases. Mani et al. (2007) discovered accurate predictors of hypertension from data in a controlled population using C4.5. Although the controlled population did not reflect the noise and sparsity of real clinical databases, they simulated this by injecting noise and found that C4.5 still performed well. Toussi et al. (2009) augmented an existing diabetes guideline with suggestions in which the guidelines fell silent, using the C5.0 algorithm on clinical data. They found that their results were similar to a newer version of the guideline, particularly regarding diabetic tri-therapy. These two applications are rather specialized, as per the limitations noted above. However, they deserve further research and appear to have applicability in workload reduction and process monitoring.

An alternate approach is *process mining*, which uses various algorithms to discover Petri nets from transaction data. Petri nets are place-transition diagrams that are popular for modeling processes. They are several steps more complicated than even a graphical model like the neural network, because they: a) have multiple 'markings' which 'move' independently around the process graph and can interact to cause transitions, and b) can

contain loops. In fact, both are central to process graphs, because they are used to discover congestion in repeating processes such as calls at a call center. Petri nets are successfully being used in healthcare to model emergency room congestion (Kopach-Konrad et al., 2007).

One popular process mining algorithm, Medeiros et al.'s (2004) α, can discover any sound structured workflow network (SWF-net) from data with no noise. (An SWF-net is a restricted type of Petri net.) However, the limitation of requiring noise-free data likely make this methodology a poor choice for mining (frequently noisy) healthcare data. Nevertheless, Vojtech and Starren (2009) are studying applicability of process mining algorithms to discover disease progression in clinical data. Their initial work, using α to discover Petri nets for patients with chronic kidney disease, did not yield human-interpretable visualizations. They concluded that the data must be more extensively preprocessed for this application of Petri nets.

1.4. Summary. A variety of studies indicate that tools used in e-commerce and other fields can be adapted to specialized applications in medical data mining. However, the only two experimental designs showed significant limitations. Wright et al. (2010) found that automated problem list maintenance is difficult in all but very tightly-correlated medication-diagnosis pairs. Klann et al.'s (2009) algorithm found useful corollary orders only about half the time. Therefore we conclude that existing, popular methodologies are insufficient to reverse-engineer decision knowledge from data.

2. Reasoning in Medical Informatics

A parallel thread of computational development in clinical medicine are physician cognitive supports. These transform expert knowledge into computational tools, most popularly for diagnostic decision support. Such tools fall broadly into two categories: knowledge-based and reasoning-based (Berner, 2009).

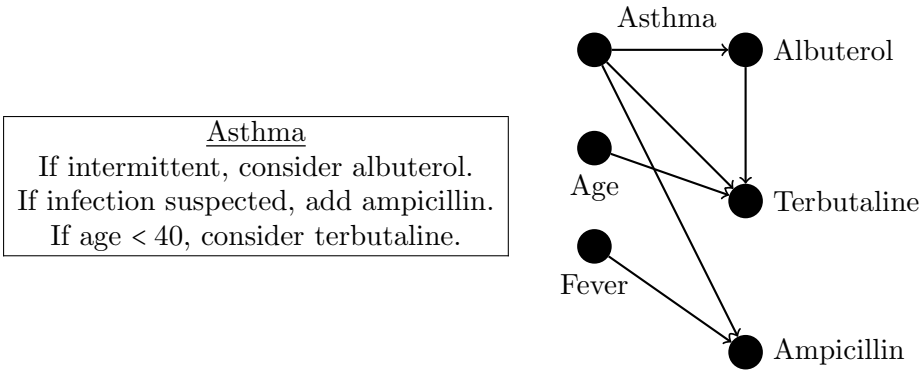
2.1. Knowledge-based Systems. Knowledge-based systems directly encode expert knowledge into some form of trigger-and-response rule. Many such rules are pairwise, such as the corollary order (e.g., *if ordering warfarin you should also place a standing order for prothombin time each morning*) (Overhage et al., 1997), alerts (e.g., suggestions of antibiotics for specific culture results), or reminders (e.g., annual mammograms are indicated for women over 40). Several languages exist in which to define these rules, many of which

were developed by the institution using them (e.g., Regenstrief's CARE language). Arden syntax is a general language for such rules (Hripcsak et al., 1994). It has guided some development but did not succeed in creating a usable national standard. The more comprehensive Guideline Interchange Format (GLIF) (Boxwala et al., 2004), which draws upon Arden syntax and popular data exchange formats (HL7 and RDF), might prove to be a more popular rule-exchange format. Recent funding activity indicates that there is now support for national repositories of such rules in a standard format, which will be an aid to both implementers and researchers (e.g., see Middleton 2009).

McDonald (1976) demonstrated that such a system can successfully change clinician behavior. He created a simple computer medication reminder system for physicians at a diabetes clinic. The system, when the physician was seeing a patient, would suggest a refill or a change in medication if either was appropriate. The study showed a behavioral change: physicians acted in the recommended way 20% more often after receiving a computerized reminder than without. This finding launched decades of study into computerized reminders and other knowledge-based systems. These have become the most popular approach to CDS and have been quite successful in inducing specific behaviors (Payne, 2000; Garg et al., 2005).

The prevalence of e-commerce approaches in clinical data mining can likely be credited to the similarity of pairwise associations to these simple knowledge-based rules. However, many decisions in medicine cannot be captured by simple trigger-and-response pairs. For example, Figure 2.2 shows a snippet of an Asthma order set from Gopher. It cannot be captured by simple pairwise rules but is better represented by a network of associations as shown on the right. This could explain why knowledge-based CDS has mostly been limited to a small set of finely tuned number reminders and alerts. More complex decision support requires a more complex approach.

2.2. Reasoning Systems. For complex reasoning, a trigger-and-response rule is often insufficient. Even in the simple example of Figure 2.2, the combinations of covariates make a rule-based representation cumbersome. Furthermore, in real clinical situations frequently not all information is known, and therefore some uncertainty would be associated with the result of the CDS.



(a) Asthma order set. **(b)** Network representation.

Figure 2.2. (Left) A snippet of Gopher's asthma order set, which cannot be represented as pairwise rules. (Right) A network of associations to represent the decision making defined by this order set.

Complex reasoning systems appropriate for situations with incomplete information have been developed in medicine since the early 1970s. These have utilized a variety of approaches to draw conclusions about specific target variables (e.g., diagnoses) given evidence (e.g., some presenting symptoms). MYCIN, developed to suggest treatments for bactermias (Shortliffe, 1976), is perhaps the most widely-known system and popularized these methods in medicine. De Dombal et al.'s (1972) tool for diagnosing abdominal pain was among the first studied in routine clinical practice. The study found that the computer selected the correct diagnosis more frequently than the senior physician on the care team, suggesting that these systems could actually be helpful to busy practitioners.

These early successes spurred a variety of other tools in the subsequent years, each of which contributed to the theory behind reasoning with uncertainty. The Quick Medical Reference (Miller et al., 1986) and its subsequent decision-theoretic version (Shwe et al., 1991), supplied diagnostic decision support in diagnosing 750 diseases internists might encounter. Heckerman and Nathwani's (1992) PathFinder specifically targeted diagnosis of lymph-node diseases.

By the 1990s, network approaches began to dominate reasoning systems. Such approaches represent all variables in a directed graph, consisting of vertices (also called nodes)

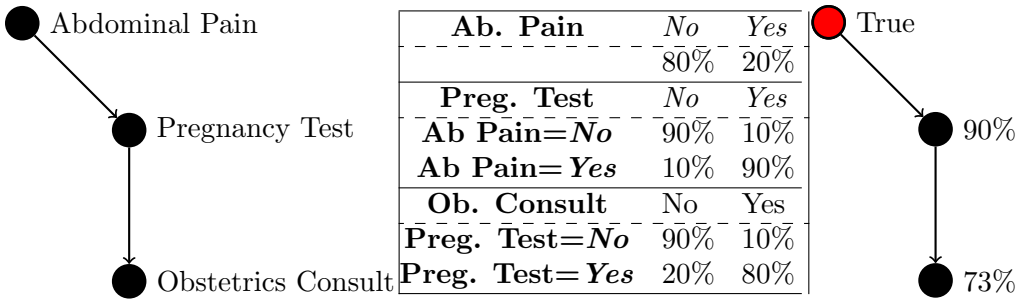


Figure 2.3. An example Bayesian Network (left), the conditional probability tables associated with it (middle), and the posterior probabilities given the evidence of 'Abdominal Pain' (right).

representing the variables and edges representing the relationships between vertices, as in Figure 2.2. These allow complex multivariate diagrams of relationships.

Methods for reasoning with these graphs have matured over the years. The basic intuition is that evidence variables can be instantiated in the network, which revises the likelihood of each target. In early systems, each node was given a score, and the edge connections were used to compute a final score based on the evidence. The Quick Medical Reference assigned scores to the edges instead of the nodes, allowing each relationship to be scored separately. Eventually network reasoning gravitated to a notion of probability, in which each node has a probability of occurring given the probability of each of its parents. As evidence is instantiated in the graph, the posterior probability of the targets gets revised. If the graph does not contain cycles, it is possible to compute the revised probabilities through 'propagation' of evidence around the graph. This is known as *belief propagation*. These probabilistic networks are known as *probabilistic graphical models*. Because Bayes' rule is used to compute the unspecified probabilities, these are also known as Bayesian Networks (BNs) or Bayesian Belief Network.

2.2.1. *The Bayesian Network.* As stated, a Bayesian network is a directed graph of vertices and edges connecting those vertices. Embedded in each node is a conditional probability table (CPT), specifying the probability of each node state given the state of each parent. An example of a simple BN, the underlying CPTs, and the revised posterior probabilities given evidence is shown in Figure 2.3.

Keeping the CPTs small is a key challenge in Bayesian network specification. It becomes quite difficult for an expert to specify the probability of a disease given an exponential number of combinations of diagnostic results. In some specialized cases, all of these probabilities are not necessary. Heckerman and Nathwani's (1992) Pathfinder system assumed that relevant symptoms independently contributed to diagnosis of a disease (e.g., nausea and a headache both increase the probability of disease independently, even though the two together might actually indicate something more extreme). With this assumption, the CPT becomes linear in number of parents, because only the contribution of each individual symptom to each disease is specified. This is known as a NOISY-OR model, a good overview of which can be found in Vomlel (2006). There are various models like this (collectively known as ICI - Independence of Causal Influence) which simplify the specification of CPTs. However, real causes are not independent of each other in most cases, and in fact one of the strengths of Bayesian networks is that causes can be interrelated. If we were to imagine a network to suggest treatments for complications of pregnancy, we would note that both hypertension and pregnancy contribute to preeclampsia, but both must be present for preeclampsia to occur. It would not be reasonable to model the impact of hypertension on preeclampsia independent of pregnancy; preeclampsia by definition only occurs in pregnant women. Therefore, a more common solution in BNs is to limit the maximum number of direct parents per node (known as the *maximum fan-in*).

Once the network is specified, the task is to determine the posterior probability of some set of targets given a set of evidence. This is the conditional probability query, and the task is known as Bayesian network inference. It is in fact intractable to directly compute these posterior probabilities in an arbitrary network. Therefore a great deal of work has gone into developing tractable approaches for inference. Pearl (1988) has been a leader in this work. He showed that in a network in which there is at most one path to any node, inference can be performed in polynomial time. This is the polytree algorithm. A natural extension of this, the junction tree algorithm, first converts the graph into a graph with few paths (by combining nodes into hybrid super-nodes) and then performs polytree inference on this hybrid graph. The *treewidth* of the graph, or the maximum number of paths in the optimal junction tree network, determines whether inference can be performed tractably.

Broadly, treewidth is higher when there are more paths between nodes in the original graph, so limiting the maximum fan-in is a good approach for inference as well (Lucena, 2003). Junction trees are the primary inference method in many popular Bayesian-network software tools (e.g., Netica).

It is also possible to pre-compile Bayesian networks into another form for faster inference. For example, Netica pre-compiles BNs to junction trees. Chavira et al. (2006) further reduces junction trees into arithmetic circuits. The compiled circuit can be reasoned over in time linear in treewidth, rather than the polynomial time of a junction tree. However, such pre-compilation might not be efficient overall, because the complexity of the junction tree is related to which variables are targets and what evidence is set (Druzdzel and Suermondt, 1994).

All of these inference approaches attempt to find an exact answer to the conditional probability query. In more complex networks, this can be impractical. Therefore approximate inference approaches, also known as particle-based inference, have also been developed. Such algorithms run simulations of 'particles' moving through the network and observe how their probabilities change. Such approaches allow inference on graphs which would otherwise be too complex.

The unfortunate endpoint of all work in Bayesian network inference is that no clever optimizations make inference tractable in an arbitrarily complex network. Cooper (1990) proved that all inference is not only intractable but is also among the least tractable set of computational problems (known as *NP-hard*). Therefore it is important when developing networks to think carefully about the complexity of the learned network.

2.2.2. *Uses of Bayesian networks in medicine.* Bayesian networks have over the years gained notoriety as the most powerful approach to medical reasoning. Unfortunately, unlike knowledge-based reminders, medical probabilistic reasoning tools have seen little use outside of academic journals. Despite the algorithmic power, limitations in network design have rendered BNs generally insufficient to model realistic, complex, and generalizable clinical situations. While it is somewhat straightforward to elicit simple rules from experts (e.g., 'if female over forty and no mammogram in the last 12 months, suggest mammogram'), it is much more difficult to extract the intricacies of complex reasoning processes. Researchers

have tackled this difficulty in various ways. One well-known example is Heckerman and Nathwani's (1992) *similarity network*, which presented domain experts subsets of the larger network, in order to simplify their thought-process in determining probabilities. Still, BNs in medicine have either been restricted to research labs (e.g., Shortliffe 1976), single locations (e.g. De Dombal et al.'s (1972) knowledge did not generalize), or instructional tools (e.g., Barnett et al. 1987).

On the other hand, Bayesian belief network methodologies have been quite successful in other fields. For example, Heckerman adapted his ideas regarding diagnostic probabilistic networks into the engine powering various Microsoft Windows help and troubleshooting systems (Hedberg, 1998). Lumiere, which first manifested in the unsuccessful Microsoft Office Assistant, still comprises the technology behind Microsoft Office Help (Horvitz, 1998).

3. Conclusion

In this chapter, we reviewed existing HIT data-mining methodologies that have been used to support decision support and found that current approaches are too weak to handle the complexity of decision support. We also reviewed reasoning methods in medical informatics, in which we focused on the Bayesian network, which (when careful attention is paid to their complexity) can rapidly reason in complex multivariate problems, given a specified model. In the next chapter, we introduce an application involving a synergy of the two domains, in which Bayesian networks are learned from data.

CHAPTER 3

A Synergy of Reasoning and Data-mining: Bayesian Network Structure Learning

We have seen that Bayesian networks are a powerful multivariate, probabilistic reasoning paradigm that have been widely studied in medicine. However, the models are often difficult to design and do not frequently generalize well and have therefore been relegated predominantly to educational and research settings. We have also seen that there is great interest in mining the large data sets which are becoming increasingly available in order to improve clinical care, but the results have thus far been meager.

It is possible to connect the power of Bayesian networks with the masses of information that data mining promises. If networks could be accurately learned from data, this would eliminate the difficulty of soliciting expert opinions on a massive number of combinations of variables. Further, this multivariate approach to data mining might be able to make sense out of the tangled correlations in clinical data.

The idea of learning Bayesian networks from data is not new. It is known as Bayesian structure learning. Already by 1997, the field was mature enough that Heckerman (1998) was able to write a comprehensive tutorial on methodologies. Bayesian structure learning has been used in medical applications, predominantly in discovering genotype-phenotype linkage (e.g., Ramoni et al. 2009). In clinical medicine, a few recent studies have used hand-constructed networks that have learned parameters (ie. CPTs) from data. Examples include Sanders and Aronsky's (2006) work on asthma detection and Hoot and Aronsky's (2005) work on liver transplant survival. However, in such cases the network structure is defined by an expert and only the probabilities are learned from data. In cases where the domain structure is not completely known, it is necessary to learn both the graph structure and the parameters from data. We are unaware of any previous studies which have done this in clinical data. Therefore we believe the use of Bayesian structure learning to mine, encode, and subsequently reason about clinical decisions is novel.

1. An Introduction to Structure Learning

The most common approach to Bayesian network structure learning (henceforth called structure learning) is a score-based approach. This begins with a set of nodes in a disconnected graph. Edges are added to choose a network that best explains a training dataset. This is typically done by maximizing a score which approximates the probability of the graph given the data, or $P(G|D)$. An approximation is used because directly computing $P(G|D)$ requires information on all possible networks (Wasserman, 2004), which is combinatorial in the number of variables in the network (Eaton and Murphy, 2007). Bayes' rule allows us to reformulate the desired probability: $P(G|D) = \frac{P(D|G)P(G)}{P(D)}$. It is permissible to ignore the prior probability of the data $P(D)$, because it is a constant given a particular dataset, and it is possible to directly compute $P(D|G)$ from the CPTs of the network. Various approximations exist for $P(G)$ (mostly designed to reduce graph complexity and overfitting). Thus, it becomes possible to find an optimal graph by maximizing a Bayesian scoring function, which is an approximation of $P(D|G)P(G)$. In this work, we use Buntine's (1991) BDeu approximation.

With a scoring function, one could theoretically search through all possible graphs to find the optimal choice. Unfortunately, because there are a combinatorial number of possible graphs ($O(n!2^{\binom{n}{2}})$) (Eaton and Murphy, 2007), this is not possible on networks of more than a few nodes. Therefore, some type of greedy search is employed to reach a local maximum. Typically, this greedy search begins with a random graph and then proceeds by adding, removing, or (optionally) reversing a single edge and re-scoring the graph. The algorithm undoes this change, but keeps track of its resulting change in score. Once all possible single changes are considered, the best local change is made. Local changes are made until the score stops increasing. In order to get closer to the global maximum (rather than the local maximum found by the greedy search), an approach from Glover's (1990) TABU framework is often utilized to diversify the search. Most commonly, when no edge change increases the score, the least negative change is made and the previous graph configuration is added to a cache. A move into one of the cached graphs is not allowed, though entries in the cache expire after some predetermined amount of time. This is a simple method to force the search to explore the search space more broadly. Another common addition is to allow a

restriction graph. Only edges in this graph are considered in the greedy search. This reduces the search space of edges if we have a priori knowledge about the domain (e.g., test results do not cause other test results, so edges between them should be disallowed). Some other strategies to reduce the search space or improve the score include: limiting the maximum fan-in (described earlier); more complex graph restrictions (e.g., requiring certain edges, allowing/restricting some edges only when some others are added, or providing a partial ordering for the nodes); or initializing the graph with an already known approximation of the correct graph. A simple greedy search implementing TABU and a restriction graph is described in Program 3.1, which we will implement in Chapter 7. It does not include edge reversals, because this slows learning and we have not seen very different graphs result from this feature. The diversification power of TABU can be adjusted by adjusting ϵ. If ϵ is 0, TABU cannot force exploration far from the local maximum; a value much less than zero increases this exploration but slows the search. In this work, ϵ is always 0, because we will primarily rely on the restriction graph to guide the greedy search.

Greedy search, like all structure learning, is rather slow. There are $O(n^2)$ possible edges to consider in each iteration, and each iteration in turn adds or removes one of $O(n^2)$ edges. Therefore, an add-only greedy search will complete in $O(n^4)$ time. However, every edge consideration is expensive. It requires partially rescoring the graph, which requires a pass through the dataset and some expensive statistical computations. Another problem is that, because the greedy search only finds a local maximum, it is common to rerun the search multiple times with random graphs as the starting point (rather than the empty graph), and to choose the best. This can substantially increase run-time.

Such restarts can be avoided through the Greedy Equivalence Search (GES), a method hypothesized by Chris Meek but then proven and formally presented by Chickering (2003). Rather than searching over Bayesian networks, it searches over what are known as 'equivalence classes' of Bayesian networks. These are Bayesian networks that all are probabilistically equivalent. If an optimal Bayesian network exists for the given dataset, GES will always find it. Due to the additional calculations needed to search over equivalence classes,

Program 3.1 Greedy-Search(D,R)

Input: R is a restriction graph. Only edges in this graph will be considered. The graph is undirected but is here treated as bidirected to simplify the algorithm.
Input: D is a a training dataset of binary variables.
G is an output graph, which initially is the empty graph.
TABU is a special queue of edges that automatically removes an edge from the queue if an attempt is made to add more than a predefined maximum.
repeat
Keep adding the highest scoring edge addition or deletion until no change improves the score.
SCORE is an array of scores for each edge in R, set to an empty array.
for all edges $e \in R$ **do**
if the reversed edge $e' \in G$ or $e \in TABU$ **then**
Do nothing - either a different edge exists between the two nodes or this edge was modified too recently.
else if $e \in G$ **then**
SCORE[e]=BDeu($G \backslash e$)
else
SCORE[e]=BDeu($G \cup e$)
end if
end for
if max($SCORE$) $\in G$ **then**
$G = G \backslash \underset{e}{\text{argmax}} SCORE[e]$
else
$G = G \cup \underset{e}{\text{argmax}} SCORE[e]$
end if
TABU $\leftarrow$ e
until max(SCORE)$\leq \epsilon$
Return G

GES is not always faster than a multi-restarting greedy search. However, its optimality property is appealing, so we will use this algorithm in the next two chapters.

Despite the speed concerns, we have found small datasets (a few thousand rows) of less than 75 variables run in a reasonable amount of time (< 30 minutes on a modern desktop computer). Therefore we will restrict our experiments in this section to small networks. In Part 2, we will extensively explore the problem of scalability.

	O_0	O_1
P_0	$\theta_{O^0\|P^0}$	$\theta_{O^1\|P^0}$
P_1	$\theta_{O^0\|P^1}$	$\theta_{O^1\|P^1}$

Table 3.1. An example abstract CPT for a node with two parents, O and P, where each variable is binary. θ is the probability of each situation.

1.1. Parameter Learning. Once the network structure is learned, the parameters in the model's probability tables (CPTs) must also be learned. Maximum Likelihood (ML) is the fastest approach, and it is appropriate when there are no missing data. In the discrete multinomial case, ML reduces to a computation for each element in the CPT of a node. For example, to compute the CPT of a node with two parents, O and P, then the entry for $O = 1, P = 1$ would be $\theta_{O^1|P^1} = \frac{M[P=1,O=1]}{\sum_x M[P=1,O=x]}$. This is just the count of rows containing both P=1 and O=1, divided by the count of rows containing P=1 (and O at any value). Because a dataset will not always have an entry for the particular combination of parents (and to smooth out empirically sampled distributions), Dirichlet hyperparameters are often introduced. These amount to adding a *pseudocount* to each count. Usually these are a uniform small value, unless some prior knowledge exists about the relationship of two nodes. Maximum likelihood can be computed in time linear with the number of CPT entries. CPT size is still exponential in the number of parents, and each entry requires a pass through the dataset, but with limited maximum fan-in this is still quite fast.

When data is missing, Expectation Maximization is used, which uses inference to 'bootstrap' ML learning. All missing values are initialized to a random value and iterations of ML (Expectation) and inference (Maximization) are performed to revise the probabilities of the missing values until they converge. This is far more time-consuming than ML, because it requires many iterations of both ML and inference. By limiting the number of iterations, a graph that can be inferred over in polynomial time can also use EM learning in polynomial time. Fortunately, for our application we can reasonably assume that there is no missing data.

2. Applying structure learning to treatment advice.

We hypothesized it would be possible to derive a small Bayesian network useful for suggesting treatments in specific situations by applying an existing structure learning algorithm to our Gopher data. A conditional probability query could then be used to determine the probability of an order given previous evidence, which could be listed in reverse probability order to produce a menu of situation-specific treatment suggestions.

We wanted to pilot this method in a complex but constrained clinical environment. Therefore we chose to examine comorbid diagnoses in hospitalizations involving pregnancy.

Program 3.2 BuildModel(D)

Input: D is a training dataset, derived from the admission-compressed Gopher table in Chapter 1.

1: Feature Selection. {We use a frequency-based selection metric to choose a fixed number of treatments and diagnoses by prevalence.}
2: Learn a Bayesian structural model. {We use Tetrad's implementation of the Greedy Equivalence Search (GES) without any graph restrictions.}
3: Estimate posterior probabilities. {We use Tetrad's Dirichlet Maximum-Likelihood parameter-estimation algorithm, with the pseudocount $\alpha = 1$.}
4: Return the learned Bayesian network.

Program 3.3 TreatmentSuggest(G,p)

Input: G is a fully specified Bayesian network of Gopher order-entry variables.

Input: p is a probability threshold. {The choice of p is application-dependent. Higher settings increase specificity and lower sensitivity. In our evaluation, we used a moderate $p = 40\%$. This threshold was chosen by experimentation. Choosing an optimal threshold is discussed in Future Directions.}

1: Manipulate known variables. {Here we set all known variables in the model to *true* or *false*.}
2: Update posterior probabilities. {Determine the likelihood of each treatment in the presence of the manipulated variables.}
3: Sort probabilities and list all treatments above a threshold p.

Pregnancy is the most prevalent chief complaint at Wishard Hospital other than pain. Also, treatments for hospitalizations involving pregnancy (delivery, complications, and postpartum care), although highly complex, involve a relatively small subset of medical orders.

We designed and implemented a method to suggest treatments for clinical situations, by deriving a Bayesian network from clinical data and computing conditional probabilities for treatment variables given known variables, using the GES and ML learning approaches discussed. We implemented our method as a series of computer programs written in SQL, Python, and Java. They utilize the free Tetrad IV suite (Ramsey, 2011) to perform Bayesian network learning and reasoning. The method has two components, the training phase (BuildModel), and the treatment suggestion tool (TreatmentSuggest). The two components are described in Programs 3.2 and 3.3.

2.1. Methods. Using the Gopher summary table developed in Chapter 1, we selected the 20 most frequent comborbid diagnoses, 30 most frequent medication orders, and 20 most frequent test orders in hospitalizations involving pregnancy. Oral analgesics are very

common, so we considered the six most frequent as a single medication in order to have a more diverse feature space. We did not filter on modality (i.e. we included both inpatient and ED visits), but we did limit our dataset to 2009 hospitalizations to restrict the data size. This resulted in 5044 hospitalizations.

We ran BuildModel on a training set of 2/3 of these data. We linked labor diagnoses which result in delivery with the diagnosis *postpartum*, because our granularity was one hospital stay, and in a successful delivery both would occur in the same stay. This resulted in 20 treatable conditions from the 20 diagnoses (we considered preeclampsia as both a labor- and non-labor condition). Using diagnoses as the known variables, we ran TreatmentSuggest for each condition with a threshold of $p = 40\%$, which produced 15 suggestion lists. Independently we asked an obstetric nurse how she would treat the 15 conditions given the 50 treatments, and we compared the results. Because our algorithm operates in an environment with existing decision support, we also evaluated which of our results might have been influenced by existing order sets.

Additionally we evaluated the network's overall ability to predict treatments in real clinical cases by calculating ROC curves for a test set of hospitalizations (the remaining 1/3 of the data). ROC curves, or receiver-operator-characteristic curves, specify the predictive ability of a system by plotting the proportion of true positives vs. false positives. The curve and the area underneath the curve (AUC) are popular methods for studying the predictive power of a system in an application-neutral manner.

2.2. Results. TreatmentSuggest did not produce treatment lists for 5 conditions because all treatment probabilities fell below 40%: false labor, vaginal bleed, vaginal discharge, abdominal pain, and high risk pregnancy. TreatmentSuggest lists for the remaining 15 conditions are presented, divided into three categories: labor and delivery (Table 3.2), problems in pregnancy (Table 3.3), and general problems (Table 3.4). Correlation with the obstetric nurse is indicated by typeface, as explained in Table 3.2's caption. This computer-nurse correlation is summarized in Table 3.5, in which we computed the precision and recall of the computer lists as compared with the nurse. Three existing order

	Treatment	Score	Indication
Common to all lists below			
∈	Epifoam	99.6%	Vaginal healing
∈	Docusate	99.5%	Constipation
∈	Analgesic	99.4%	Pain & healing
∈	IV Fluids	98.6%	Throughout delivery
∈	**Oxytocin**	**94.8%**	**Prevent hemorrhage**
			Induce delivery
∈	*Simethicone*	*90.8%*	*Post-surgery gas*
∈	Lanolin	87.1%	Breastfeeding
∈	*Prenatal Vitamin*	*66.5%*	*Prenatal*
	Syphillis Screen	*65.3%*	*Prenatal*
	Blood Typing	64.3%	Postpartum bleeding
∈	Morphine Injection	61.9%	Pain
Preterm Labor & Postpartum			
∈	**Betamethasone**	**74.7%**	**Steroid for baby's lungs**
∈	*Strep B Probe*	*68.5%*	*Could infect child*
∈	*Chlamydia Screen*	*62.9%*	*Potential cause*
∈	*Ferrous Sulfate*	*41.4%*	*For anemia*
∈	*Penicillin*	*41.2%*	*For infection e.g., strep B*
Preeclampsia & Postpartum			
	Mg Sulfate	**74.6%**	**Prevent eclampsia**
	Kidney Tests	58.4%	Routine for preeclampsia
	Liver Tests	49.1%	Routine for preeclampsia
Active Labor & Postpartum			
Early Stage Labor & Postpartum			
	Nothing additional.		

Table 3.2. TreatmentSuggest lists for labor-related conditions. The first list shows treatments that appeared in all lists (with nearly identical probabilities). Subsequent lists show additional treatments for specific conditions. Typeface corresponds to an independent evaluation of the feature set by an obstetric nurse. Bold is her opinion of the most important treatment for the condition, plain text was indicated, and italics were not indicated (but not necessarily clinically inappropriate – many are routine disease screens). A symbol to the left of the treatment indicates decision-support order sets exist and include this treatment.

sets overlapped with the 20 conditions: postpartum, preterm labor, and back pain. A set membership symbol indicates the order is also found in one of these sets.

2.2.1. *Predictive Ability.* We calculated ROC curves for all treatment variables on a test set of pregnancy hospitalizations. Table 3.6 shows the highest, lowest, and average AUC values. We also show two characteristic ROC curves from among the worst (Figure 3.1) and best (Figure 3.2). Finally, to visualize the connections in the graph, we display the Markov

Treatment	Score	Indication
Abortion Threatened		
Pregnancy Test	**71.9%**	**Always done prior**
Pyelonephritis		
Ceftriaxone	**76.4%**	**Antibiotic**
Nitrofurantoin	45.1%	Antibiotic
Yeast infection		
Chlamydia Screen	*59.4%*	*Potential cause*
Fluconazole	**49.8%**	**Antifungal**
Bacterial Vaginosis		
Metronizadole	**87.7%**	**Antibiotic**
Chlamydia Screen	*60.8%*	*Potential true dx*
Urinary Tract Infection		
Nitrofurantoin	**63.3%**	**Antibiotic**
Urine culture	*40.5%*	*Used to diagnose*
Hypertension		
Liver tests	**52.9%**	**Routine in pregnancy**
Kidney Tests	**47.1%**	**Routine**
Preeclampsia		
Liver tests	**46.3%**	**Routine**
Kidney tests	**45.4%**	**Routine**

Table 3.3. TreatmentSuggest lists for complications of pregnancy. Typefaces are as described in Table 3.2 (corresponding to an obstetric nurse's evaluation). Kidney and liver tests are both key treatments in two cases, so both are bold. Note that there was no overlap here with existing order sets.

blanket (a node's children, parents, and siblings) for two diagnoses. Figure 3.3 produced a short and Figure 3.4 produced a long treatment list.

2.3. Discussion. TreatmentSuggest captured accurate and non-trivial clinical knowledge in all 15 suggestion lists. The five conditions with no list were all indicated by the nurse as not directly treatable (i.e. more information was needed). All 15 lists contained the key treatment for the condition (selected by the obstetric nurse from among our 50). Examples of key treatments include steroids for preterm labor and magnesium sulfate for severe preeclampsia. In all but one non-labor condition, the key treatment was the first (3 cases) or only suggestion (7 cases). Nearly all of the treatment suggestions appear to be indicated, either mentioned specifically by the nurse (71% precision) or still having a specific

Treatment	Score	Indication
Nausea With Vomiting		
Promethazine	**58.9%**	**Nausea**
Esophageal Reflux		
Ranitidine	**89.6%**	**Stomach trouble**
Back pain		
∈ **Analgesic**	**48.8%**	**Pain**
Fever		
Analgesic	**57.5%**	**Fever reduction**

Table 3.4. TREATMENTSUGGEST lists for general conditions. Typeface and symbols are as described in Table 3.2 (corresponding to an obstetric nurse's evaluation and existing decision support content, respectively.)

Computer vs. Nurse	
Precision	71%
Recall	77%

Table 3.5. Summary statistics of the computer-to-nurse agreement demonstrated in Tables 3.2, 3.3, and 3.4

Treatment	AUC	Uses
Best		
Kidney & Liver Tests	99.5%	Many problems
Bupivicaine	99.8%	Epidural
Oxytocin	99.1%	Delivery & Postpartum
Naloxone	99.0%	Reverse narcotic
Betamethasone	99.0%	For baby's lungs
Worst		
Promethazine	68.3%	Nausea
Ranitidine	67.1%	Stomach
Azithromycin	65.87%	Antibiotic
Ondansetron	61.4%	Nausea
Average	*87.3%*	

Table 3.6. The pregnancy network's predictive ability on a test set of hospitalizations, shown as the best and worst AUCs for treatment variables. Many high AUCs were pregnancy-specific and most low AUCs were not. Average AUC for all 50 treatments was .873.

use (e.g., a strep B probe could be indicated for a woman about to deliver if she has had no prenatal care). This lists selected many of the treatments independently suggested by the

Figure 3.1. The ROC curve for oxytocin,which had one of the highest AUCs (.991). It is used almost exclusively in labor.

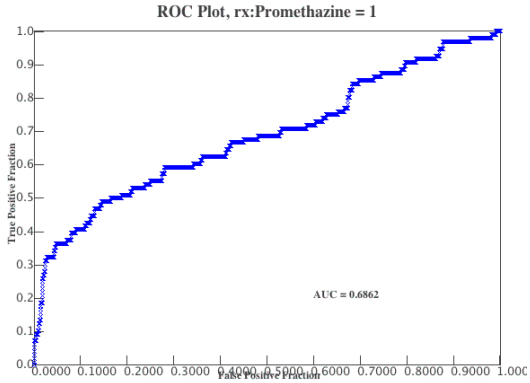


Figure 3.2. The ROC curve for promethazine, which had one of the lowest AUCs (.683). It is a non-specific nausea drug.

nurse (77% recall). Additionally, the network demonstrated strong ability to predict treatments (average AUC .873), and all treatments that have a specific use in pregnancy-related conditions had an AUC above .9, with the highest value being the epidural bupivicaine (.998). The lowest AUC value was .614 for the common anti-nausea drug ondansetron.

Our Bayesian network approach successfully created a model of multivariate relationships in the data, and, using only diagnoses as the basis for prediction, it produced strong results. This demonstrates Bayesian networks with structure learning are a promising new approach for harnessing the 'wisdom of the crowd' in decision support. Including more treatment context (e.g. current treatments and combinations of comorbid diagnoses) has the potential to further improve the accuracy of the treatment lists.

In addition to our primary application of treatment suggestion lists, the results indicate this method could also be used for compliance monitoring. The suggestion lists for postpartum care, preterm labor, and back pain parallel existing order sets, which surprised us because order sets must be called up manually by name and are generally believed to be infrequently used. This approach could be used to discover the effectiveness and usage of existing order sets, the nature of divergence from them, and inappropriate treatments.

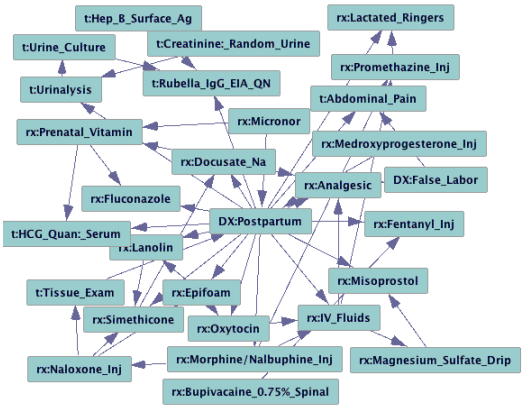


Figure 3.3. The Markov blanket of the network for *postpartum*, which correlated with longer suggestion lists.

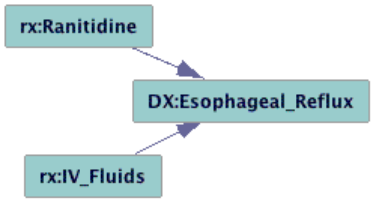


Figure 3.4. The Markov blanket of the network for *esophageal reflux*, which correlated with shorter suggestion lists.

2.3.1. *Limitations.* This study involved a small, constrained subset of medicine and therefore its performance might not be indicative of the general case. Evaluations in other areas of medicine are necessary.

The use of a single practitioner for a reference standard is insufficient to reliably validate all clinical knowledge. For example, the nurse did not indicate prenatal vitamins for postpartum women, even though this is routine and appears in the order set. Additionally, she has not practiced in our environment, where many patients do not have insurance and therefore receive more disease screenings during labor than women with a standard course of prenatal care. This highlights that practitioners themselves do not necessarily reflect the local standard. Further evaluation should include a larger and preferably local reference group.

Finally, this method, as is true of this entire work, relies entirely on local data. Therefore it automatically produces localized treatment suggestions, but it also captures both the best practices and the bad habits of physicians. Best practices might be better approximated by selecting a subset of physicians for training the model. However, this approach will never entirely eliminate the need for expert-developed (or at least expert-validated) decision support content.

2.3.2. *Conclusions.* Here we have demonstrated a computational method that produces treatment suggestions for conditions using local order-entry data and Bayesian networks. In a dataset of hospitalizations involving pregnancy and 70 frequent order-entry features (50 treatments and 20 diagnoses), our method produced treatment suggestion lists for 15 conditions. The lists captured accurate and non-trivial clinical knowledge, and all contained the key treatment for condition, according to an independent evaluation by an obstetric nurse. Outside of labor-related situations, that key treatment was the first or only suggestion 90% of the time. On a test set of pregnancy-related hospitalizations, the Bayesian network generated by our method predicted treatments with an average AUC of .873 and predicted pregnancy-specific treatments with even higher accuracy (AUC above .9).

Although this pilot study evaluated a method on a small subset of medicine (with strong clinical correlations between conditions and treatments), this type of method shows promise in producing general 'wisdom-of-the-crowd' decision support content. Bayesian networks make it possible to produce highly-tailored suggestions for very specific patient situations.

3. Future Directions in this Dissertation

This study lays a promising groundwork. It also highlights many areas for improving the basic methodology, which outlines the rest of this dissertation.

- TreatmentSuggest uses only a single comorbid diagnosis as context. As mentioned, more treatment context might improve results. (Chapter 4)
- BuildModel uses a simple frequency-based feature-selection heuristic, but the most frequent are not necessarily the most important features. (Chapter 6)
- Our model is quite small, and including a larger set of treatments and diagnoses would allow reasoning about a wider variety of situations. (Chapter 7)
- Our method does not incorporate temporal relationships within the hospital stay. Therefore it is impossible to generate a treatment list for 'postpartum' separately from 'labor', because both occur within one stay. This is why we see both lanolin (for sore nipples after breastfeeding) and betamethasone (a steroid used in preterm labor) in the same list. Techniques do exist for incorporating temporal reasoning into Bayesian networks. (Chapter 8)

- Including some non-treatment variables (such as demographics and key test results) would provide a fuller picture of patient context. (Chapter 9)
- The standard conditional probability query might not always be optimal. In particular: a) less critical treatments are sometimes shown higher in the lists than more critical ones, because the score is a probability of ordering (e.g. epifoam is listed higher than oxytocin); and d) our algorithm used a manually selected threshold of 40% which might not be optimal. These weaknesses would be aided by developing a modified query for 'expected decisions.' (Chapter 9)
- The evaluation was small and involved only a single practitioner who did not practice in our environment. A more comprehensive evaluation is needed. (Chapters 4 and 10).

CHAPTER 4

A refined system and evaluation measures

In the previous chapter, we developed a computational method that produces treatment suggestions for conditions using local order-entry data and Bayesian networks. In a dataset of hospitalizations involving pregnancy and 70 frequent order-entry features (50 treatments and 20 diagnoses), our method produced treatment suggestion lists for 15 conditions. The lists captured accurate and non-trivial clinical knowledge, and all contained the key treatment for each condition, according to an independent evaluation by an obstetric nurse. Outside of labor-related situations, that key treatment was the first or only suggestion 90% of the time. On a test set of pregnancy-related hospitalizations, the Bayesian network generated by our method predicted treatments with an average AUC of .873 and predicted pregnancy-specific treatments with even higher accuracy (AUC above .9).

TreatmentSuggest generates a single list given a clinical situation, but our goal is to evaluate treatment suggestion lists as a clinical situation evolves. In this chapter, we: build a revised recommendation system that responds dynamically to suggest the most common next orders based on what has been ordered previously, develop a novel evaluation methodology to determine how well our system reproduces reasonable behavior in the cases under study, and apply this methodology to the system in four domains.

Program 4.1 TreatmentSuggestInteractive(G)

Input: G is a Bayesian Network Model
E is a set of evidence, such that $E \subset v \in G$, where initially $E = \varnothing$.
repeat
Update beliefs {Compute the posterior probability of all $v \in G \notin E$.}
Create a list of all $v \in G \notin E$ in descending order of posterior probability, stopping at an optional threshold.
Display the list to the user and wait for the user to choose an order from the list.
Add the selected order from to E.
until the user closes the session.

1. A New Methodology

1.1. TreatmentSuggestInteractive. This new methodology, TreatmentSuggestInteractive, is summarized in Program 4.1. We implemented this methodology in Java using the SMILE toolkit (Druzdzel, 1999), a freely available toolkit for network inference. We

also wrote a converter in Java to translate the Tetrad network output by BUILDMODEL into SMILE format. A prototype of this interface can be seen in Figure 4.1. In the interface, the user can iterate between placing orders or observing diagnoses and then generating a suggestion list based on that current situation.

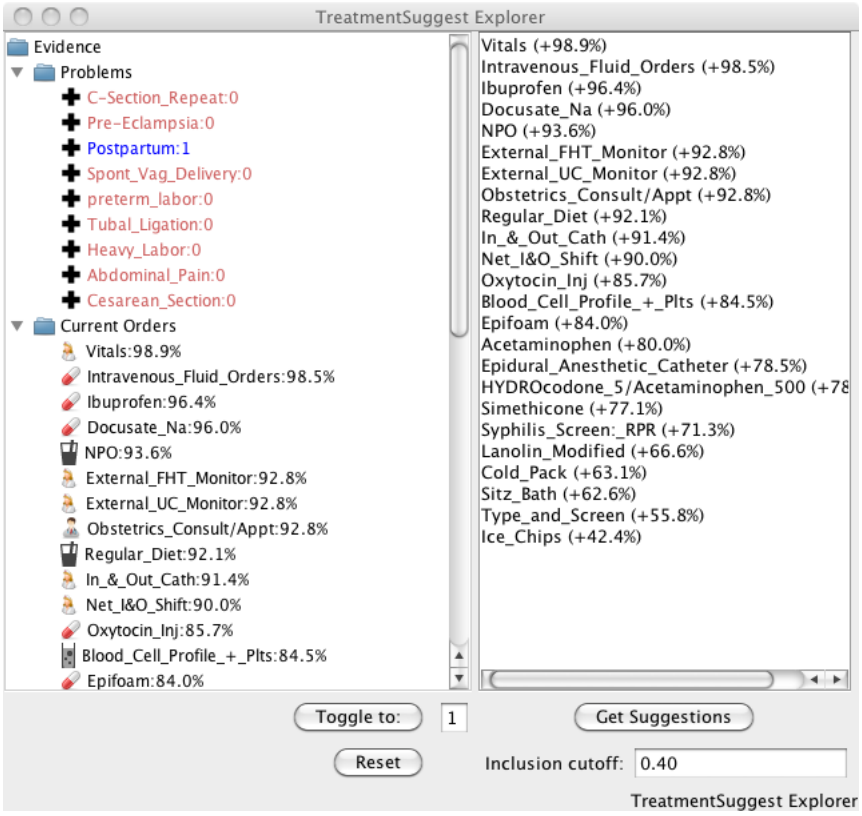


Figure 4.1. A prototype implementation of TREATMENTSUGGESTINTERACTIVE. The left pane shows the current evidence and possible treatments, arranged in descending order of probability, as defined by the network given the evidence. The right pane shows a summary of those treatment suggestions above a threshold of 40%. The toggle and reset buttons allow the user to change the evidence.

1.2. Evaluating simulated hospitalizations. In Chapter 3, we used the AUC of an ROC curve of the network's predictions to compare our network's inference to local practice patterns. However, this is a measure of the BN and not a treatment suggestion methodology. To measure how frequently lists generated by Program 4.1 suggest the correct next order, we developed a simulation approach motivated by Program 4.1. We run an automated version of Program 4.1, generating a suggestion list after every order in each encounter

and measuring where in the suggestion list the next order appears (if at all). We can then generate aggregate statistics of the predictive ability of the system by order, to measure to what degree the situation-specific lists correctly list each order.

One straightforward statistic to measure the value of suggestion lists might be the positive predictive value (PPV) of the system in predicting each order. However, it is possible to reach a PPV of 100% by choosing an arbitrarily long list, so we fix our desired PPV to 80% and report the length of list required to reach that PPV. We call this PPV80 for shorthand. This statistic has a commonsense meaning: *How long must our list be in order to usually contain the next order?*. We chose a desired PPV of 80% because our experiments show generally consistent behavior about 80% of the time followed by a very long tail of variation. This is a reasonable choice because our system's goal is to capture the consensus of the data, not the variation among individual practitioners.

We also can compute the AUC of an ROC curve of our methodology. This is the area under a plot of the true positive rate vs. false positive rate for each order in the suggestion list in the session in which it is ordered. This also has a commonsense meaning: *the probability that during the session in which an order is placed, it will be ranked higher than in previous sessions.* We label this the tAUC, to highlight that we are measuring the AUC of our temporal-simulation methodology.

To compute PPV80, we must record where in the menu each order occurred when it was selected in each hospitalization. We store this in $\vec{M}_o$, a set of frequency arrays indexed by suggestion list menu position. These store the number of times each order o appeared at each menu position, at the time it was first selected in each hospitalization. Computing PPV80 involves finding the lowest index of $\vec{M}_o$ at which 80% of all cases have occurred.

We can directly compute tAUC without first computing the ROC curve via the Wilcoxon test of ranks (W), which, remarkably, exactly corresponds to the AUC of an ROC curve (Hanley and McNeil, 1982). A definition of the Wilcoxon statistic is presented in Table 4.1, similar to the presentation by Hanley and McNeil (1982). As shown in the table, it requires two vectors for each order type o: $\vec{T}_o$, a list of probabilities of true positives (i.e.

$$W(\vec{T},\vec{F}) = \frac{\sum_t^{\vec{T}} \sum_f^{\vec{F}} S(t,f)}{\|\vec{T}\| * \|\vec{F}\|} \qquad S(t,f) = \begin{cases} 1 & \text{if } t > f \\ 0.5 & \text{if } t = f \\ 0 & \text{if } t < f \end{cases}$$

Table 4.1. The Wilcoxon test of ranks, which can be used to compute the area under the receiver-operator curve. $\vec{T}$ is a list of posterior probabilities for true instances (in a test set) of a particular order, and $\vec{F}$ is the corresponding list for false instances.

the probability when o is the next order chosen); and $\vec{F}_o$, a list of probabilities of false positives (i.e. before o is chosen).

This simulation and calculation methodology is summarized in Program 4.2, which we implemented this algorithm in Java using SMILE.

2. Evaluation

We chose to evaluate this methodology on four modalities of medicine in Wishard: inpatient medicine, the emergency department (ED), ED visits triaged to the urgent visit clinic (UVC), and the intensive care unit (ICU). Each modality reflects different aspects of medicine. Inpatient care focuses more on treatment than diagnosis in a longer-term stay, the ED involves a shorter stay involving both diagnosis and treatment, the UVC involves a very brief 'stay' focused on diagnosis, and the ICU involves tightly-correlated actions for very specific care.

2.1. Methods. We moved from our four selected modalities to four domain-specific BNs in the following steps:

(1) We focused our domains on the most frequent diagnosis / complaint for the first three modalities: visits involving pregnancy in inpatient medicine, back pain in the ED, and hypertension in the UVC. For the ICU, we chose the Medical ICU (MICU).

(2) We selected visits from our Gopher summary table corresponding to each domain. This involved 6192 ED back pain, 1214 UVC hypertension, 3229 inpatient pregnancy, and 709 MICU visits.

Program 4.2 Eval(G,D)

Input: G is a Bayesian Network Model
Input: D is a test dataset of hospitalizations, consisting of orders and co-occurring problems by session.
$\vec{T}_o$ is a set of arrays to contain probabilities of orders when they are the next choice.
$\vec{F}_o$ is a set of arrays to contain probabilities of orders before they are the next choice.
$\vec{M}_o$ is a set of frequency arrays to contain counts of menu positions for orders when they are chosen.
for all hospitalizations in D **do**
  $E = \varnothing$ {Clear the evidence in the network }
  for all order sessions in the hospitalization **do**
    for all orders o not in E **do**
      Set evidence for all orders and co-occurring problems that have occurred up to this point in the hospitalization (including this session).
      Create an array of posterior probabilities, $\vec{L}$, for all orders that are not already in E and arrange in descending probability order.
      if o did not occur in this session **then**
        $\vec{F}_o \leftarrow \vec{L}[o]$ {Record the probability of the order in the list of false instances.}
      else if o occurred in this session **then**
        $\vec{T}_o \leftarrow \vec{L}[o]$ {Record the probability of the order in the list of true instances.}
        Increment $\vec{M}_o[\text{index}(\vec{L}[o])]$ by 1 {Record the depth in the menu at which this action occurred.}
        $E \leftarrow o$ {Add o to the evidence.}
      end if
    end for
  end for
end for

for all orders o **do**
  $AUC_o = W(T_o, F_o)$
  $PPV80_o$=the first index of the accumulation of M_o that is > 80% of the total accumulation.
end for

(3) For nodes in each network, we selected the 40 most frequent orders in each domain that occur on average less than twice within a stay (because our system suggests each order only once)[1]. We also included up to the 10 most frequent co-occurring diagnoses and complaints (though sometimes fewer than 10 diagnoses/complaints co-occurred with the primary problem). These were used in our BN learning

[1]This actually excluded a minority of orders, because orders that frequently recurred in long stays also tended to occur in short stays, thus lowering the average.

and evaluation as evidence, but were not presented in the suggestion menus (because the goal is to suggest treatments, leaving diagnosis to clinicians). The diagnoses/complaints used in our networks can be seen in Table 4.2.

(4) We split each data set into a training set (2/3 of admissions) and a test set (1/3). We used admission-compressed data in the training set (i.e. a single row per admission), because the GES algorithm does not support temporal data. In the test set we retained the time-series of order sessions, because this is consistent with applying simulation of Program 4.1 to hospitalization data.

(5) To generate BNs for each domain, we used the approach of Program 3.2 (i.e. the Greedy Equivalence Search with Maximum Likelihood estimation and Dirichlet hyperparameters initialized to $\alpha = 1$, using the Tetrad toolkit).

(6) We wrote an exporter to convert the three networks from Tetrad into SMILE format, which is the format required by EVAL (Program 4.2).

Pregnancy, Inpatient		**Back pain, ED**		**Hypertension, UVC**		**Medical ICU**	
Postpartum	89%	Car Accident	4%	Med Refill	27%	Sepsis	3%
Cesarean Section	4%	Neck Pain	3%	Diabetes Mellitus	16%	Abdominal Pain	3%
Spont Vag Delivery	2%	Abdominal Pain	3%	Back Pain	6%	Hypotension	3%
Tubal Ligation	1%	Chest pain	2%	Abscess	6%	Dysphagia	3%
Pre-Eclampsia	1%	UTI	2%	CAD	4%	Unspecified Surgery	2%
Preterm Labor	1%	Headache	1%	Toothache	4%		
Abdominal Pain	1%	Knee Pain	1%	Cellulitis	4%		
C-Section Repeat	<1%	Hypertension	1%	Headache	3%		
Failed Induction	<1%	Med Refill	1%	COPD	3%		
		Shoulder Pain	<1%	Hyperlipidemia	<1%		

Table 4.2. The co-occurring diagnoses and complaints in each domain-specific network, listed by their prevalence in the test sets. These were used as evidence as they appeared in the test cases, and were not part of the predictive evaluation.

2.2. Results. We learned the four domain-specific Bayesian Networks: inpatient pregnancy, back pain in the emergency department, hypertension in the urgent visit clinic, and the medical intensive care unit. Each network contained 40 orders, except the urgent visit network where only 31 orders co-occurred with hypertension. Also the networks contained up to 10 additional co-occurring diagnoses/complaints, which are listed in Table 4.2.

We ran our implementation of Program 4.2 on the four networks with their corresponding test set, the results of which we report here. Summary statistics can be seen in Table 4.3 (average tAUC and average PPV80, weighted by the frequency of each order). Two histograms, one showing tAUC and the other showing PPV80, can be seen in Figure 4.2. For each domain, the 10 orders in which the system performed best and worst (by PPV80) are shown in Table 4.4.

Domain	**W.Avg. tAUC**	**W.Avg. PPV80**
Inpatient Pregnancy	.844	3.94
Back pain in the Emergency Department	.765	6.98
Hypertension in the Urgent Visit Clinic	.741	6.11
Medical Intensive Care Unit	.714	13.34

Table 4.3. For each domain, the weighted average tAUC (area under the receiver-operator curve in time-series data) and PPV80 (minimal menu length that positively predicts 80% of orders). Weighting is by frequency of order.

Order name, tAUC, and PPV80 for each domain

Pregnancy, Inpat.			Back pain, ED			Hypertension, UVC			Medical ICU		
Sitz Bath	0.99	0	Pelvis CT	0.99	0	Protime	0.95	0	Mg Level	0.90	0
Cold Pack	0.99	0	Abdomen CT	0.99	0	Clotting Test	0.99	0	Urine Culture	0.92	0
Naloxone	0.99	0	Low Spine X-ray	0.68	0	Blood Profile	0.88	1	Urinalysis	0.86	1
Lung Exercises	0.99	0	Lat. Chest X-ray	0.66	1	Cardiac Mrkrs	0.99	2	Phos. Test	0.86	1
UC Monitor	0.99	0	Blood Cell Profile	0.95	1	Urinalysis	0.81	3	Monitor I&O	0.92	1
Ibuprofen	0.98	0	EKG	0.88	2	Metabolic Panel	0.73	3	SCD	0.85	3
FHT Monitor	0.97	0	Metabolic Panel	0.87	2	Med. Consult	0.69	3	Vitals	0.86	3
Docusate Na	0.96	0	Vag. Infect. Test	0.93	2	Urine Drug Test	0.92	3	Blood Culture	0.81	4
Monitor I&O	0.94	0	Cardiac Markers	0.96	2	Metabolic Panel	0.77	5	P.T. Consult	0.87	5
Promethazine	0.88	10	Hip X-ray	0.79	23	Dental Consult	0.82	18	Lactate	0.52	25
Foley Catheter	0.91	11	Low Spine MRI	0.64	24	Blood Culture	0.93	21	Hang Blood	0.74	26
IV Lock	0.73	11	Chlamydia Test	0.78	28	TSH	0.50	23	Cardiac Echo	0.64	27
Transfusion	0.65	14	Knee X-ray	0.62	30	P.T. Consult	0.57	23	Tylenol	0.69	29
Ice Chips	0.72	15	Sport Consult	0.60	32	Knee X-ray	0.49	25	Speech Therapy	0.63	30
Nalbuphine	0.83	16	EPIC Referral	0.59	33	Neuro. Consult	0.13	25	Albuterol	0.47	31
Pfizerpen	0.76	24	Shoulder X-ray	0.68	34	Uric Acid	0.74	26	Head CT	0.43	33
NPO	0.75	26	Low Spine CT	0.63	36	T4 Test	0.50	27	Blood Gas Test	0.51	34
Oxytocin	0.68	26	Drug Urine Tst	0.67	36	Head CT	0.56	28	Central Cath.	0.45	34
Morphine	0.50	28	Neuro. Consult	0.68	37	Hgb A1C	0.72	30	Aspirin	0.54	34
Urine Drug	0.71	30	Wrist X-ray	0.61	39	Ophth. Consult	0.73	31	Hydralazine	0.50	36

Table 4.4. Order name, tAUC, and PPV80 of the ten best and worst order predictions in each domain. 'Best' and 'worst' are chosen by PPV80, a measure of the minimal menu length that displays 80% of the orders at the time they actually occurred in the test cases (lower is better). Temporal AUC (tAUC) is also reported (higher is better).

2.3. Discussion. The evaluation of our treatment suggestion system on four domain-specific BNs against test cases drawn from the same environments showed fairly strong overall performance. In particular, our treatment suggestion lists can be short (3.94-6.11 items capture 80% of all orders) in three out of four networks. Also, the system ranks orders higher in the session they are ordered than prior to ordering, with high probability (71%-84%).

More insight can be gained from the differences across domains. For example, while the inpatient pregnancy model performed very well overall (weighted average tAUC .884 and PPV80 3.94), the MICU network did much worse (weighted average tAUC .714 and PPV80 13.34, respectively). We suspect this reflects the amount of additional context needed to predict orders in each domain. For example, in the pregnancy network, a diagnosis of postpartum might be sufficient to lead to the next treatments (e.g., various adjuncts like

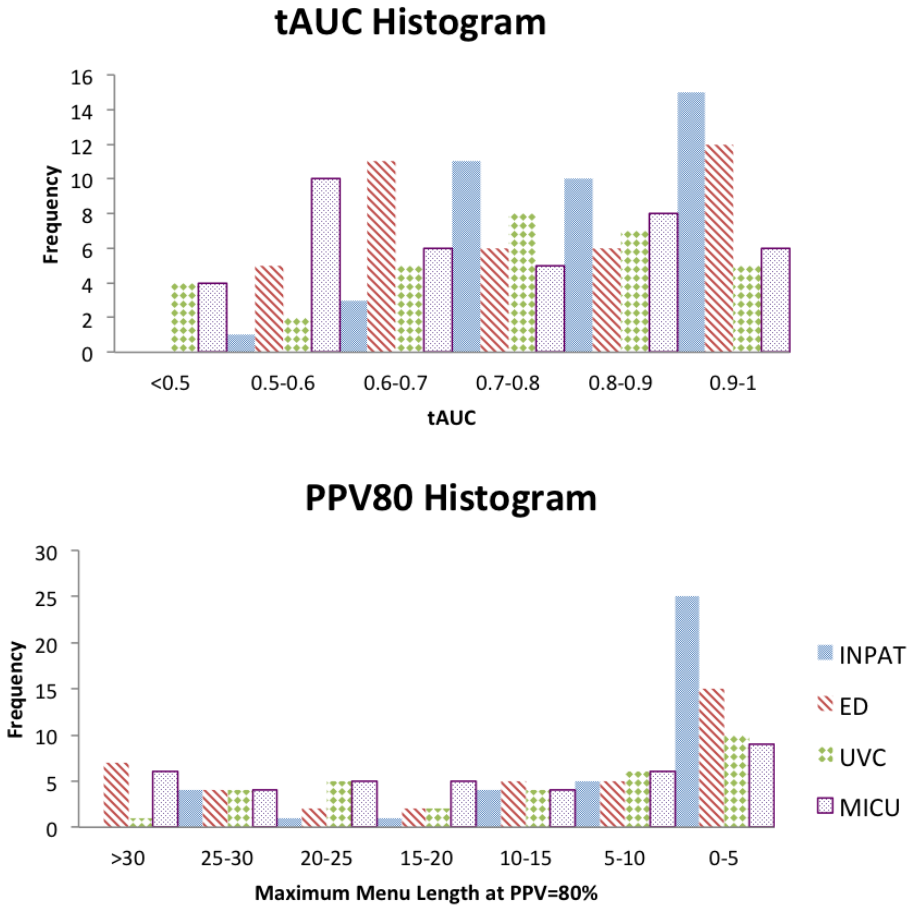


Figure 4.2. Histograms for each order in each network (see legend). (Top) The temporal area under the receiver-operator curve (tAUC). (Bottom) The treatment suggestion menu length given a positive predictive value of 80% (PPV80).

epifoam), but in the MICU, test results and external information are more likely to be important (e.g, is the patient on a ventilator?). Moreover, in the MICU, the co-occurring diagnoses/complaints were very rare, making the available context even less substantial. Finally, our system is designed to suggest treatments, and it does not currently suggest diagnoses. This might reflect the poorer performance in environments where diagnosis is the primary goal (e.g., the UVC).

Also interesting are the differences among orders within a domain. Although some orders are suggested almost exactly when they should be (e.g., a cold pack in pregnancy visits and a pelvis CT in the ED, for example), others appear at the bottom of long lists and

are not predicted much better than chance (a neurology consult in the ED, for example). We again suspect that poorly performing orders are missing the context needed to predict them. In some cases, the missing context could be other orders or diagnoses that could serve as proxies (e.g., a head CT might serve as a proxy for suspected neurological injury leading to a neurology consult, for example). This points to the need for more principled feature selection.

A final interesting discovery is that tAUC is less strongly correlated with PPV80 than we expected. Although frequently good predictors (high tAUC) are at the top of our menus (high PPV80), this is not always the case. Both alternatives can be seen in Table 4.4. A lumbar (lower) spine X-ray in the emergency department has high PPV80 but low tAUC, and a nalbuphine order in inpatient pregnancy has low PPV80 but high tAUC. In the first case, we expect the order stays at the top of the list until it is picked because it has a high prior probability. In the second case, we suspect that although nalbuphine's probability increases when it is actually ordered, it is never high enough to outweigh other orders. This leads us to believe that choosing order-specific probability thresholds might be appropriate.

2.3.1. *Limitations.* This new system improves on the one in Chapter 3 with dynamic menus and evaluation metrics that simulate real clinical situations, but otherwise it has all of the limitations mentioned in the previous chapter. These include: frequency based feature selection, small networks, no temporal learning, lack of non-treatment context, and the potentially non-optimal conditional probability query. All of these will be addressed in future chapters.

2.4. Conclusion. In this work, we have expanded our Bayesian-network based treatment suggestion system to dynamically recommend the most common next orders based on what has been ordered previously. Additionally we have developed a novel time-series evaluation methodology to determine how well our system reproduces reasonable behavior, which we applied to four domains. These included inpatient pregnancy, abdominal pain in the emergency department, hypertension in the urgent visit clinic, and the medical intensive care unit.

We found our system performed fairly well in all domains but had a variance which directly suggested areas for improvement. It performed best in inpatient pregnancy (weighted

average tAUC .844, weighted average menu length at PPV 80% 3.94) and worst in the medical intensive care unit (weighted average tAUC .714, weighted average menu length at PPV 80% 13.34). There was also variance within domains, ranging from near-perfect performance (cold pack in inpatient pregnancy) to very poor performance (hydralazine in the medical intensive care unit had tAUC .51 and menu length at PPV 80% 36). We hypothesize that higher performance correlates with a greater number of factors needed to predict the order than are present in the network.

We have now established a methodology to translate CPOE data into an interactive treatment suggestion tool and we have developed two novel measures to evaluate its accuracy compared with real hospitalizations (Programs 3.2, 4.1, and 4.2). We have found performance if fairly strong in small networks. In the following section, we introduce various novel methodological improvements to structure learning and inference which will further improve our methodology.

PART 2

Novel Methods for Bayesian Network Learning in Large Domains

CHAPTER 5

Bayesian Networks in Large Domains

1. Introduction

In the previous part of this dissertation, we developed a novel approach to augment decision-support content using local data and Bayesian network learning. We concluded with a system that performed reasonably well in creating problem-specific treatment suggestions in several domains. However, the networks contained at most 70 variables which were chosen using a simple frequency heuristic and about 5000 hospitalizations. Inpatient medicine is much larger. Specifically, Gopher alone includes 7376 order and problem variables and approximately 100000 hospitalizations per year. The RMRS contains another 3033 test result types, and there are various additional demographic and encounter metadata one might like to add. At present, we have no methodology to deal with this much larger domain.

1.1. One network to rule them all? A compelling idea is to learn a single network to capture the entire environment of inpatient medicine. Unfortunately, this is not possible, for at least three reasons.

1.1.1. *Large-sample network learning is NP hard.* First, Chickering et al. (2004) have shown that the general problem of large-sample learning of Bayesian network structure is NP-hard. This conclusion makes intuitive sense when we imagine the number of different networks that can exist for a given number of variables. In fact, the number of possible networks is super-exponential in the number of variables: $O(n!2^{\binom{n}{2}})$ (Eaton and Murphy, 2007). Therefore, as we discussed in Chapter 3, some type of heuristic search is usually employed to discover a 'good' graph. Unfortunately, heuristic searches are still (in the general case) NP-hard.

Nevertheless, much work has been done in developing heuristics that perform well in particular situations. However, they cannot tractably learn a network in a domain as broad as inpatient medicine. As a practical example, one of the fastest current learning algorithms (Max-Min Hill Climbing) took 13 days to learn a network for a 5000 variable by 5000 sample dataset on a 2.4GHz Pentium Xeon (Tsamardinos et al., 2006). We will explore in more detail the computational limits of structure learning algorithms in Chapter 7, but practically

network learning is currently limited to networks of less than a few hundred variables when the sample-size is large, and this is still time-consuming.

1.1.2. *Probabilistic inference is NP hard when treewidth is unbounded.* Second, as was discussed previously, the general problem of probabilistic inference is also NP-hard (Cooper, 1990). We also discussed how practical speed of inference is highly correlated to the width of the optimal junction tree decomposition, or *treewidth* (Lucena, 2003). More paths between nodes in the original network can increase the treewidth. Frequently, increasing the number of variables will increase the number of paths, which will in turn increase the treewidth of the network. This is not the case if the network naturally separates into subnetworks, but in our experimentation in Gopher data, most common orders are correlated with many others (for example, IV fluids is correlated with most everything). These 'hub' nodes result in increased treewidth as the number of nodes in the graph increases. Therefore the number of variables in the network must remain limited.

		Betamethasone	
Preterm Labor	**Epidural**	Present	Absent
Present	Present	0.77	0.33
Absent	Present	0.10	0.90
Present	Absent	0.53	0.47
Absent	Absent	0.01	0.99

Table 5.1. The training data must be partitioned into a number of partitions exponential in number of parents. Shown here: a CPT of a binary node with two binary parents, which has 2^2 CPT entries.

1.1.3. *The training data must be partitioned into a number of partitions exponential in number of parents.* Third, recall that each node in a Bayesian network is defined by a conditional probability table (CPT), which defines the probability of that variable in every combination of its parents. These tables grow exponentially in number of parents p. For example, in a network of binary variables, the number of entries in the CPT is 2^p. Each entry in the CPT requires the training data to be *partitioned*. Therefore we need a dataset that is large enough to support the number of partitions of the largest CPT in the network. Commonly, a minimum of ten samples is required for each partition (Tsamardinos et al., 2006; Spirtes et al., 2000), however empirical data is uneven and noisy, so the number of

entries per partition must be larger. With binary variables and a minimum partition size of twenty, then, the maximum number of parents as a function of the number of samples is: $p = \log_2(s/20)$. Thus the number of possible edges in the network is proportional to the square of the number of variables, while the number of possible parents is proportional to the logarithm of the number of samples. This means that adding variables to a network requires an exponential increase in the number of samples in the training set to support the same number of parents.

To contextualize, Beinlich et al.'s (1989) classic ALARM network has 37 nodes and 46 edges. This is 6.9% of possible edges. In contrast, a 5000 variable network could suppport $\binom{5000}{2}$ = 12497500 edges, but with 25,000 samples (the approximate number of inpatient encounters in Wishard per year), $p = 10$ and the maximum number of edges is 50000. This is less than .4% of possible edges!

Therefore, full Bayesian networks must involve smaller, domain-specific networks. However, it is possible to gain some understanding of the domain through other means. Here we introduce what we will call the Maximal Association Graph (MAG). The MAG will contain a superset of edges in the actual Bayesian network, and it will become an essential tool for both choosing and learning domain-specific networks in subsequent chapters. In order to develop the MAG, we must first understand D-separation.

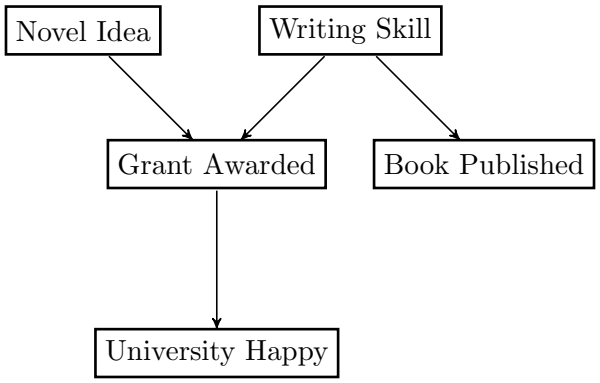


Figure 5.1. A network representing the life of a university investigator. The university is happy only when grants are awarded.

1.1.4. *D-separation.* D-separation (dependency separation) is a statement about whether a pair of nodes A and B are *independent* given an optional set of evidence E, denoted D-sep$(A; B|E)$. If two nodes are not D-separated, they are D-connected.

The example in Figure 5.1 shows a simple network that distills the life of a university investigator. The university is happy only when a grant is awarded. However, University Happy is also indirectly connected to many other nodes, even those as far distant as Book Published. This is because getting a book published increases the likelihood that the investigator has good writing skills, which increases the probability of a grant being awarded, which in turn makes the university happy. All nodes in this network are D-connected to University Happy. It is in fact true that two nodes can influence one another if and only if they are D-connected given some evidence.

Moreover, with no evidence, the only two nodes which are D-separated in this network are Writing Skill and Novel Idea. This seems intuitively strange because Writing Skill and Novel Idea both influence the grant being awarded. However, knowing nothing about a grant being awarded, knowing an investigator has an awful idea says nothing about their writing skill. If the investigator did have a novel idea and did not get a grant awarded, then we might believe that their writing skills are poor. Therefore, D-sep$(W; N)$ is true, but not D-sep$(W; N|G)$. This is an example of a *common cause*, which is the only situation in which adding evidence D-connects nodes. Two nodes related because they are common causes are called *colliders*.

In all other situations, adding evidence will D-separate nodes. For example, if we know that a grant has been awarded, then we know the university will be happy whether or not the investigator has good writing skills, had a novel idea, or had a connection in the granting agency whom they blackmailed into getting a grant. Therefore, D-sep$(U; *|G)$ (where $*$ refers to all other nodes).

This leads to several important conclusions.

THEOREM 1. *If* $\exists\{S\} \in V$: D-sep$(A; B|S)$*, then there is no edge between* A *and* B*.*

THEOREM 2. *If* D-sep$(A; B)$*, then either there is no path between* A *and* B *or they are only on a path involving colliders (e.g., the only path involves a common effect).*

By Theorem 1, D-separation relationships completely define what is called the *graph skeleton*, which is the undirected version of the Bayesian network. Constraint-based learning algorithms take advantage of this, which we will explore in two chapters. By Theorem 2, a graph defined by first-order D-separation (the case with no conditioning set) directly links all nodes that have a path between them not involving colliders. This is both a superset of the edges in the final network and a good measure of the positive correlations between nodes (as in this example, common causes often have a negative correlation). One final theorem is necessary to define the MAG:

THEOREM 3. D-sep$(A; B|E) \equiv A \perp B|E$ *(D-separation in a graph is the same as statistical independence in a data set.)*

This useful property means that, we can perform a statistically sound independence test on the training set to determine D-separation. Throughout this work, then, we will use D-sep$(A; B)$ interchangably with $A \perp B$.

1.1.5. *The Maximal Association Graph.* We define the MAG as the first-order D-separation graph discussed above. Specifically, the set of edges in the MAG is defined as:

$$\forall (A, B) \in \{V\} : (\text{A,B}) \notin E \equiv \text{D-sep}(A; B)$$

In the subsequent two chapters, we will use these properties for feature selection and to speed learning. Here we will develop a very rapid technique using SQL to learn this superstructure on sets of binary variables. We will focus on binary variables because our order-entry data from Chapter 1 is binary (either an order/diagnosis is active or it is not), but it is possible to generalize this methodology to discrete multinomials as well.

2. Maximal Association Graphs in Relational Databases

It is straightforward for a computer to learn a Maximal Association Graph (MAG) in memory. By Theorem 3, it requires $\binom{n}{2}$ conditional independence tests, one for every pair of variables. Unfortunately, because each conditional independence test requires a pass through all the data, this can become extremely slow in large datasets. Therefore, we leverage the power of relational databases, which are highly optimized for finding overlap and

performing aggregate operations on massive datasets. If an algorithm can be reformulated as joins and intersections of sets, relational databases are frequently much faster than an iterative approach.

This explains the growing number of association algorithms that are implemented directly in SQL (e.g., Klann et al. 2009; Sidl and Lukcs 2006). However, we are not aware of any tools which use relational databases to support Bayesian structure learning. There is work to modify relational databases to support probabilistic queries (Wang et al., 2010), as well as packages to perform simple statistics on databases[1]. None of these can be used directly to learn the structure of Bayesian networks.

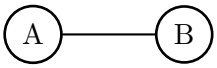
Determining Bayesian network structure requires exponentially more statistical computations than association rules. This can be understood through a contingency table. For a binary variable, the four cells in Table 5.2 must be known in order to check for statistical independence (and therefore D-separation, by Theorem 3). An association rule is only concerned with the top left cell (shown in bold). This difference is again shown in Figure 5.2, which defines a relationship between A and B as both an association rule and a D-separation relationship. However, set algebra will allow us to reduce this additional computation dramatically, so that we can compute D-separation nearly as quickly as an association rule directly in SQL.

	B=1	B=0
A=1	$\boldsymbol{M(A \cap B)}$	$M(A \backslash B)$
A=0	$M(B \backslash A)$	$M(\overline{A \cup B})$

Table 5.2. A contingency table for two binary variables, where M() is a function that counts the members of the sets. Shown in bold is the only cell considered in association rules.

2.1. Methods: Computing the Maximal Association Graph. Here we develop a rapid method for computing the Maximal Association Graph in SQL. SQL's strengths are counting occurrences and finding the overlap of sets, so we reformulate the contingency table in Table 5.2 into overlaps and counts using set algebra. If two variables A and B overlap, their contingency table can be visualized as the Venn diagrams in the top half of

[1]http://poststat.projects.postgresql.org/

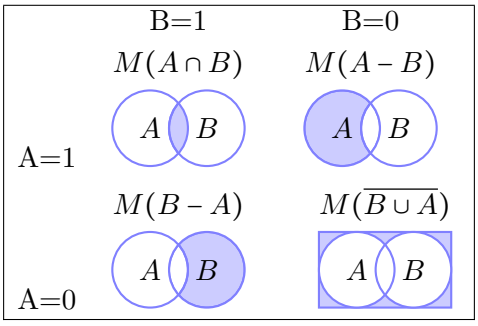


(a) Relationship of A and B.

Context	Definition
Association rule	$\text{confidence}(A,B) > t = \frac{O_{11}}{E_{11}}$
Bayesian Network	$\text{D-sep}(A;B) = A \perp B \approx 2\sum_{ij} O_{ij} \ln(O_{ij}/E_{ij})$

(b) Meaning, in terms of observed (O) and expected values (E).

Figure 5.2. A comparison of the meaning of a simple network (left) as defined by association rules and Bayesian networks. The indices of O and E relate to the column and row numbers of the contingency table (see Figure 5.2).



	B=1	B=0
A=1	$M(A \cap B)$	$M(A) - M(A \cap B)$
A=0	$M(B) - M(B \cap A)$	$M(T) - M(A) - M(B) + M(A \cap B)$

Figure 5.3. Transforming a two-variable contingency table for binary variables into the magnitudes of overlaps, using axioms of probability.

Figure 5.3. The quantities in the Venn diagrams lead to the modified contingency table in the bottom half of Figure 5.3 through probability axioms [2]. This new contingency table requires three counts and one set overlap computation (the totals of A, B, the universe, and the overlap of A and B), all of which a relational database can calculate very quickly. If A and B do not overlap, our modified set algebra simply does not include the $M(A \cup B)$ terms. Once the contingency table is calculated, a statistical independence test (such as the one shown in Figure 5.2) determines whether an edge exists between nodes.

A SQL program that uses this modified contingency table to compute the MAG is illustrated in Program 5.1. In step one, the counts and overlap are computed for all cases

[2]The axioms can be found at http://mathworld.wolfram.com/ComplementSet.html

Program 5.1 MAG(T). T is a SQL table of transaction ids (tid) and orders (o). Shown here is the relational algebra corresponding to the SQL code for computing the MAG. Step 1 calculates the counts of each variables and the overlap of their pairs. It then inserts all pairs with a G^2 value above 3.54 into the edgelist. Step 2 tests the G^2 value of edges without overlap through an antijoin.

Step 1: $A \cup B \neq \varnothing$

$$
\begin{aligned}
TOT \leftarrow & \Pi_{count}(\mathcal{G}_{count(*)}(T)) \\
A \leftarrow & \Pi_{o,count}({}_{o}\mathcal{G}_{count(tid)}(T)) \\
AB \leftarrow & \Pi_{o1,o2,count}(\sigma_{o1<o2} \\
& ({}_{o1,o2}\mathcal{G}_{count(tid1)}(\rho_{T1(tid1,o1)}(T)) \bowtie (\rho_{T2(tid2,o2)}(T)))) \\
E \leftarrow & \Pi_{o1,o2}(\sigma_{G(a,bt,abt,count)>3.54} \\
& (TOT \times \rho_{o1,o2,abt}(AB) \bowtie \rho_{o1,at}(A) \bowtie \rho_{B(o2,bt)}(A)))
\end{aligned}
$$

Step 2: $A \cap B = \varnothing$

$$
\begin{aligned}
E \leftarrow & \Pi_{o1,o2}(\sigma_{G(a,bt,0,count)>3.54} \\
& (TOT \times \bowtie \rho_{o1,at}(A) \bowtie \rho_{B(o2,bt)}(A) \triangleright AB)))
\end{aligned}
$$

in which A and B overlap, considering only edges in the ordering $A < B$ (because the edges are undirected, this actually considers all edge combinations). Edges are then added to the edge list E if a statistical independence test rejects independence. In this case, the G^2 test is used (see next paragraph). In particular, an edge is added if the G^2 value exceeds 3.54, which is the critical value for one degree of freedom. Step two calculates the G^2 value of the remaining edges where $A \cup B = \varnothing$ using an antijoin. In sparsely connected spaces, this step is much slower. Note that both steps could have been combined through an outer join, but we separate them for clarity.

Program 5.1 uses the G^2 test. Although any statistically valid independence test can be used, we choose the G^2 test because, although it is asymptotically equivalent to X^2, it errs on false positives rather than false negatives on small samples (Larntz, 1978). It is generally preferable to retain a false edge when finding network skeletons, because once the edge is eliminated it cannot be added again. Therefore G^2 has become the test of choice for Bayesian structure learning (Spirtes et al., 2000).

G^2 is related to the likelihood-ratio of variables and is defined as follows (Fienberg, 2007):

$$G^2 = 2 * \sum_{s \epsilon \{S\}} Observed * \ln(\frac{Observed}{Expected})$$

where degrees of freedom are, as in X^2, computed as:

$$df = (Cols - 1) * (Rows - 1)$$

3. Evaluation

In the following two chapters, we will explore the utility of the MAG in choosing and learning Bayesian networks. Here we compare the speed of our SQL approach to a standard iterative in-memory approach.

3.1. Methods. We implemented Program 5.1 in PostgreSQL 8.4 (MAG) directly in SQL, and we implemented G^2 as a SQL stored procedure. The output is a table of pairs of ids representing edges. For our reference in-memory implementation, we used Tetrad's Fast Adjacency Searcher version 4 (FAS4), which also checks for independencies using the G^2 test. We configured FAS4 to perform exactly the same independence tests as MAG (by default, it finds further independencies through conditional G^2 tests).

We used the admission-compressed inpatient database table from Chapter 1. We kept 2/3 of this data as a training set, which we used in this analysis. This set contained 5,358 variables (problems and orders) and 44,860 admissions. We will refer to this as the GopherInpatient set.

As a secondary evaluation, we used GEnie (part of a network inference toolkit created by Druzdzel (1999)) to generate 10,000 data instances from Conati et al.'s (1997) Andes network, a well-known Bayesian network involving only binary variables, used for physics tutoring. We will refer to this as the Andes set.

To perform speed tests, we wrote a Java program to randomly permute the variables in each set and select subsets of increasing size. In each iteration, the program generates a data subset containing all samples but only the variables in the current iteration. It feeds this to MAG and FAS4 and compares the speed. The speed test begins after the data subset is created and all data structures are set up, so only the runtime of FAS4 and the MAG script are compared.

We ran this Java program on the dataset using an increment of 250 variables, with a maximum time limit of 6 hours.

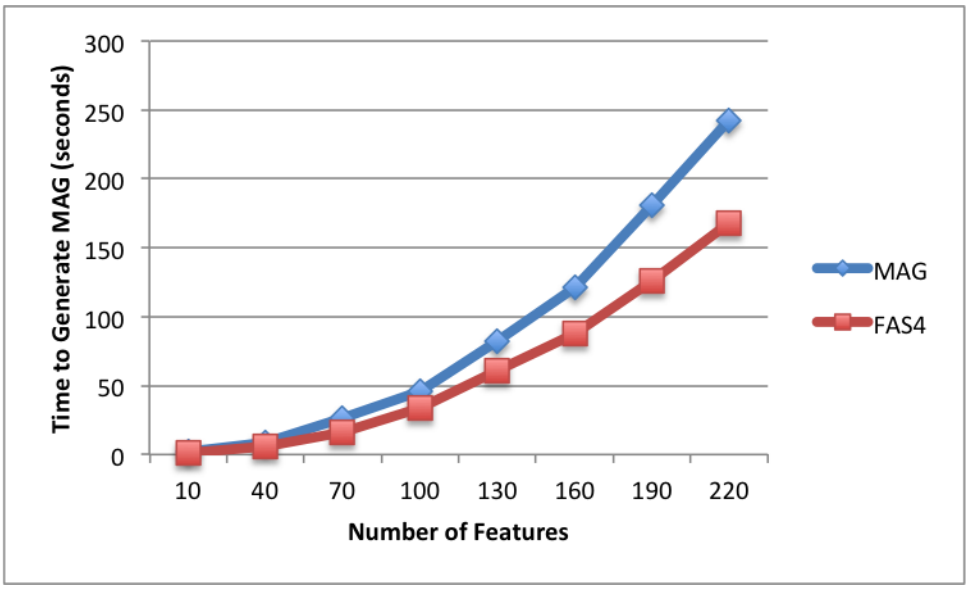
3.2. Results. Figure 5.4 compares the speed of FAS4 versus MAG (up to 3000 variables) on the Andes and GopherInpatient datasets. The full MAG for GopherInpatient contained 664,029 edges. The 100,000 edges with the highest G^2 value are shown in Figure 5.5. The MAG for Andes contained 274 edges.

3.3. Discussion. SQL shows a massive speed improvement over Tetrad in GopherInpatient, including a nearly-logarithmic growth pattern (compared to the nearly-exponential growth pattern of FAS4). However, SQL is somewhat slower than Tetrad on Andes. We suspect this has to do with the density of the domain.

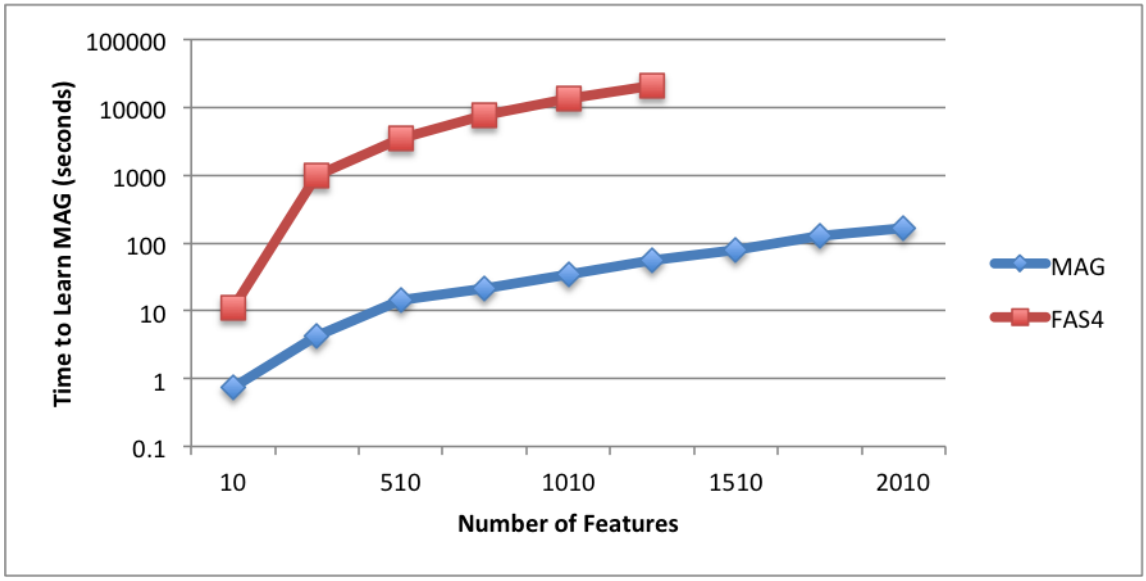
Tetrad uses an efficient algorithm to compute G^2 that requires $v * r$ in-memory reads of the input dataset, where v is the number of variables (e.g. two plus any conditional restrictions) and r is the number of records in the database. Therefore the number of reads is not dependent on the number of positives.

In relational databases, the set-theoretic notion creates very large intermediate tables when joining sets with large amounts of overlap, but it can detect non-overlapping sets very quickly. The Andes dataset had 980,362 positive instances and GopherInpatient had 1,423,634. In terms of positive instances, the entire Andes datset was about half of the size of the entire GopherInpatient set, even though it has 25 times fewer variables.

Therefore, we conclude that in-memory implementations tend to scale as a function of $v * r$, whereas database implementations tend to scale as a function of positive instances. Therefore, the sparser the dataset, the greater win SQL will provide. Therefore the set-theoretic optimizations of relational database make it optimal for learning a superset of



(a) Andes dataset. (linear scale)



(b) GopherInpatient dataset (log scale).

Figure 5.4. The performance of MAG vs FAS4 for two test datasets.

Bayesian network structure in sparse domains. In two chapters, we will show that this translates quite directly into speed increases in learning full Bayesian network structure. First, however, we will demonstrate how to use the MAG for principled feature selection.

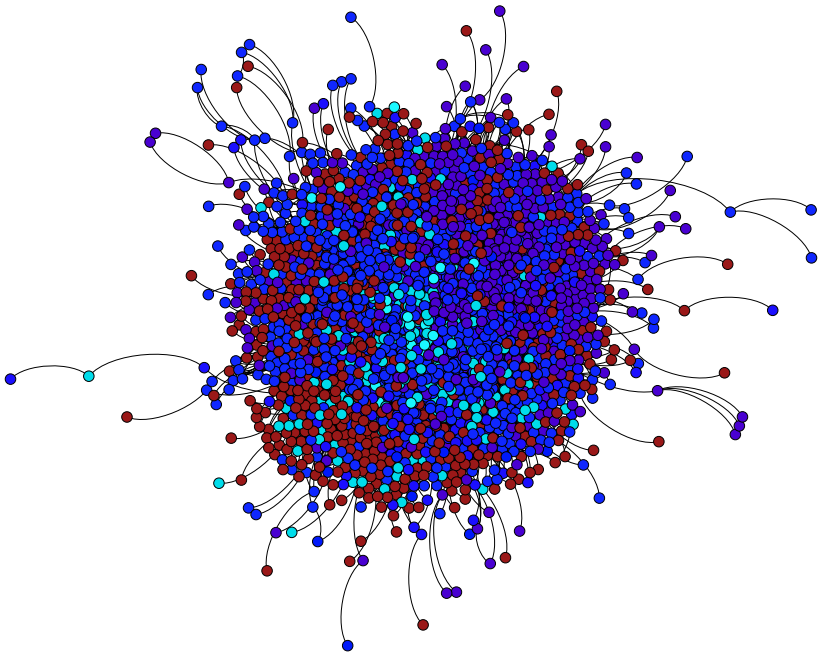


Figure 5.5. The 100,000 edges in the Maximal Association Graph for the GopherInpatient dataset with the highest G^2 value. The full graph contains 664,029 edges. Problems are colored red, tests are purple, nursing orders are cyan, and all other orders are blue.

CHAPTER 6

Principled Feature Selection for Networks of Association

1. Background

In the previous chapter, we examined three reasons it is not possible to have one network for all of medicine. Therefore we must rely on *feature selection* for choosing the most important variables in smaller domain-specific networks. In Chapters 3 and 4, we used a simple frequency heuristic, but frequency is not highly correlated in general with importance. For example, IV flushing is very frequent but might not be important, and uterine hemorrhage is uncommon but has a great deal of importance in some clinical situations.

1.1. The *all relevant* problem. Finding the most-relevant variables in a dataset is not a common problem in machine learning. Rather, much of machine learning focuses feature selection on finding the *minimal optimal* set of variables to predict a target, e.g. 'What is needed to predict whether the patient has diabetes?' There are well-established methods of finding the *minimal optimal* feature set. For example, Tsamardinos and Aliferis (2003) have shown that a set of variables known as the *Markov Blanket* are the most relevant variables in classifying a variable.

However, the alternative question, 'What does one do with a diabetic patient?', is less explored. It involves many more variables, because it asks for associations, not predictors. Nilsson et al. (2007) call this the *all-relevant* problem. They develop recursive Markov Blanket variable selection algorithms to choose this. However, recursively selecting the Markov Blankets of even a single target variable eventually selects nearly all possible variables. This has the same problem as the association rule studies we explored in Chapter 2, in which nearly everything has some association with everything else. Furthermore, recursively discovering Markov blankets on large datasets is computationally intractable. This has approximately the same complexity as discovering the network skeleton, which in turn is a loose upper bound for complexity of discovering the entire Bayesian network, which Chickering et al. (2004) have shown is NP-hard. (We will discuss network skeletons in more detail in the next chapter.) Because feature selection deals with truly large sets of variables (such as the 7376 Gopher order entry variables), recursive Markov blankets are not a viable solution.

One might think that limiting the recursion depth in Nilsson et al.'s (2007) recursive Markov Blanket algorithm would choose the most associated variables. After all, such variables would be 'closer together' in the final network. However, Koller and Sahami (1996) demonstrate that 'closeness' in a Bayesian network is not a strong measure of association. Therefore, we turn to heuristics.

The simplest heuristic is frequency of co-occurrence, used in Part 1. As mentioned, frequency does not correlate strongly with relevance, and so it leaves something to be desired.

Perhaps the next level of complexity is association rule mining and collaborative filtering, discussed in Chapter 2. This actually solves a feature selection problem. For example, Amazon.com's recommendations (Linden et al., 2003) show the most-relevant shopping recommendations. Although, as discussed, in complex situations association rules are still not a good choice for *choosing* features, we will show they provably *eliminate* irrelevant features.

Much closer in robustness to Bayesian network reasoning (but without the computational complexity) is network analysis, which applies various graph-theoretic measures to analyze the relative importance of nodes in a given graph. In this motif, associations (such as protein-protein interactions) are arranged into a network. Each edge is given a single weight (and so these networks are much less complex than the CPTs of Bayesian networks). We can apply network analysis to the MAG (Chapter 5) to produce a relevant set of features.

In this chapter, we develop a two-pass feature selection algorithm that eliminates variables with association rule mining and then applies network analysis to refine the feature set, using a MAG generated with the remaining variables.

2. An Algorithm for Principled Feature Selection.

2.1. Association rule mining. In Chapter 2, we noted that association rule mining did not do well in capturing the most important interactions in medicine, unless those associations were very strong and specific. However, it is mathematically certain to remove features that have no interaction with the target.

Confidence, a common and frequently-used measure of association, is defined as: $\frac{P(A \cap B)}{P(A)}$, or the probability that seeing A means also seeing B in the same sample. If A and B never

co-occur, its confidence is zero. The bidirectional version, *all-confidence*, measures the probability that both A implies B or B implies A, or $\max(\frac{P(A \cap B)}{P(A)}, \frac{P(A \cap B)}{P(B)})$.

This of course only measures positive interactions, whereas Bayesian networks measure all forms of variable interaction (see Chapter 5). However, to choose an initial set of features this is exactly what we want. For example, in medicine *prostate cancer* might interact with *pregnant*, in that it is slightly more likely to be true when pregnant is false. However, adding *prostate cancer* to a network focused on pregnancy would clutter our network with uninteresting associations.

Therefore we use association rule mining not to *recommend* features, but only to *eliminate* features with a very low all-confidence. Specifically, we choose one or more target variables to define our domain (for example, 'pregnancy') and find all variables that have an all-confidence with any of the targets greater than some very small number ϵ. This algorithm is illustrated in Algorithm 6.1.

Program 6.1 Feature-Assoc(D,T)

Input: D is a data set of transactions, consisting of items in a transaction
Input: T is a set of target variables that define our domain
O is an output set of selected features, which is initially empty.
for all $v \in D \notin T$ **do**
  for all $t \in T$ **do**
    if all-confidence$(t, v) > \epsilon$ **then**
      $O \leftarrow v$ {Choose the feature}
    end if
  end for
end for
return O

Note that this is different than eliminating features by a frequency heuristic. For example, a tension headache is only recorded once in all Gopher inpatient hospitalizations, in a pregnancy hospitalization. Therefore, even though it occurs less than .01% of hospitalizations (and co-ocurrs with pregnancy only .02% of the time), it has a confidence that 'tension headache implies pregnancy' of 100%. Therefore at this stage it is retained. Conversely, even though a hepatic function panel occurs in 10% of inpatient hospitalizations, its *all-confidence* with pregnancy is 0.6%.

2.2. Markov Importance. The second phase in our feature selection utilizes the Maximal Association Graph (MAG) developed in the previous chapter to refine the feature set chosen by Program 6.1. As discussed, the MAG contains an edge for all active paths between variables that do not involve a collider. If we weight the MAG by the edge's G^2 values, we have an association network appropriate for network analysis. Unlike in a full Bayesian network, all variables relevant to a target are within a short number of links to the target. Specifically, the maximum distance of a relevant variable is the number of colliders k that might separate the target from a relevant variable, when the only paths between the variables involve a collider. In practice, we have found that k can be very small. As an example, a good guess for k might be the average path length between two nodes in the network. In a MAG generated for pregnancy from GopherInpatient the average path length is just 2.2. (We will discuss actually choosing k at the end of this section.)

We will take advantage of these properties of the MAG to develop a method of choosing a set of relevant variables through analysis of 'walks' through the graph. The same idea lies behind Google's search engine.

2.2.1. *PageRank and its variants.* Google's PageRank algorithm ranks search results with a score defined as follows (Page et al., 1999): *for a random surfer on the internet, how likely is it that they will spend time at a given page?* To compute this, PageRank views the entire Internet as a directed graph, where nodes are webpages and links are edges. Starting with some initialization of ranks, it defines a recursive algorithm that propagates ranks around the graph. Each webpage has a ranking proportional to the sum of the ranks that point to it, adjusted for the number of links on the source page.

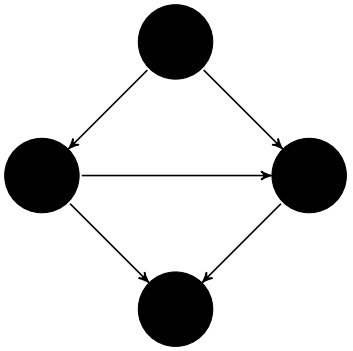
The PageRank of a page p can also be viewed as a 'random walk' in the web graph. A random walk in a graph is a Markov Chain, which is a state-transition process which specifies a starting state for all nodes (in the case of PageRank, an arbitrary vector of postive numbers) and transition probabilities to all outbound nodes. In PageRank, the transition probabilities are uniform across all links on a page, so each link on page i has probability $1/N_i$, where N_i is the number of links on page i. This state-transition process can be represented compactly as a vector ρ_0 of starting states and a matrix of transition

probabilities A. Figure 6.1 shows a sample network with the associated matrix of transition probabilities A.

It is then a matter of matrix multiplication to find ρ at time 1: $\rho_1 = A\rho_0$. This can be performed iteratively to find the probability that the surfer is at any page at any given time. Therefore PageRank can be computed as the sum of such multiplications over all time, as shown in Equation 1. Figure 6.2 shows the propagation of PageRank through the network in Figure 6.1.

THEOREM 4. *Given a vector of starting states ρ_o and a matrix of transition probabilities A, then the likelihood of being in any state at time n is specified by $\rho_n = A^t\rho_0$.*

DEFINITION 1. *PageRank $= \sum_{t=0}^{\infty} A^t\rho_0$, where the summation halts when R reaches convergence*[1]*. ρ_0 can be almost any vector; the same relative PageRank will be produced.*



$$A = \begin{bmatrix} 0. & 0.5 & 0.5 & 0. \\ 0. & 0. & 0.5 & 0.5 \\ 0. & 0. & 0. & 1. \\ 0. & 0. & 0. & 0. \end{bmatrix}$$

Figure 6.1. A sample network structure and the PageRank transition probability matrix associated with it, defined by Theorem 4.

PageRank has since been applied to various feature selection problems, but one of its weaknesses is that it estimates a global importance, e.g. the steady-state rank given that the 'surfer' is equally likely to start anywhere. White and Smyth (2003) have developed new algorithms that extend the concept of PageRank to determine relative importance, e.g. the rank given that the surfer will start from one among a set of root nodes.

Their k-step Markov importance algorithm limits the sum in Definition 1 to k steps and uses ρ_0 to define the root nodes (whose rank is evenly distributed among the root nodes

[1]This slightly simplifies the algorithm. The actual algorithm also accounts for random jumps and sinks.

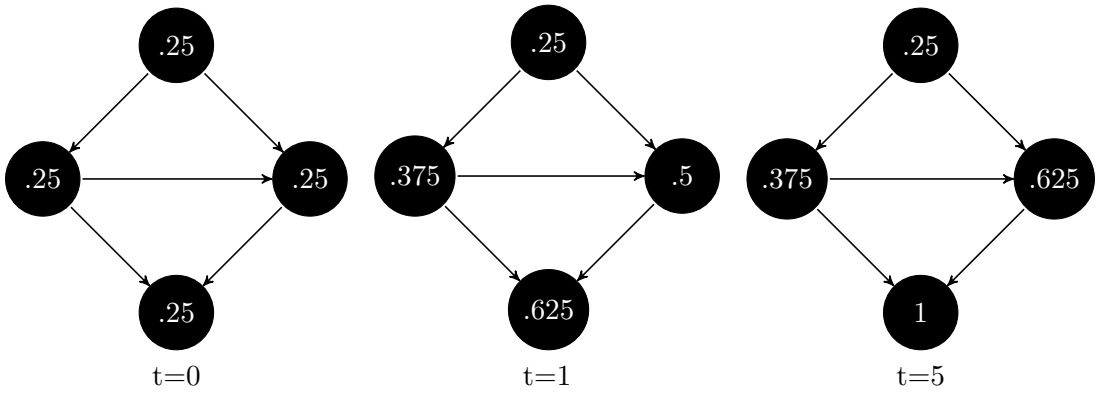


Figure 6.2. A sample propagation of PageRank, as described in Definition 1. Left is the (arbitrary) starting state, middle is after one iteration, right is after convergence (which is five iterations in this case). Nodes are labeled by their rank.

and 0 otherwise). Therefore, k-step Markov importance is defined as Markov Importance = $\sum_{t=0}^{k} A^t \rho_0$. This biases PageRank to walks of a fixed length from the root nodes, so as $k \to \infty$, K-step Markov importance → PageRank. Further, rather than using uniform probabilities in the transition matrix, arbitrary weights can be specified, so weighted graphs (which are common in much practical graph analysis) are directly supported. A sample run of k-step Markov Importance on the same network in Figure 6.1 can be seen in Figure 6.3.

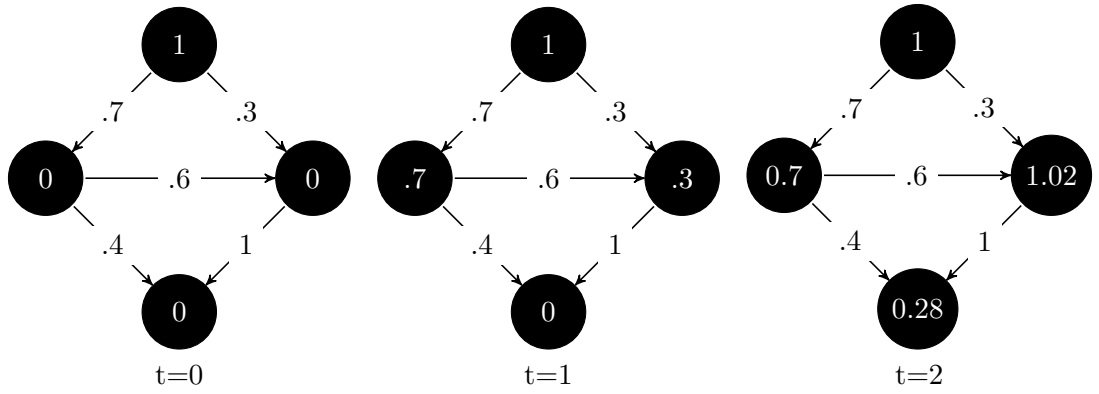


Figure 6.3. A sample k-step Markov Importance with $k = 2$. Normalized edge weights are shown on the graph. Left is the starting state, middle is after one iteration, right is after two iterations. Nodes are labeled by their rank.

2.2.2. *Principled feature selection.* Therefore we use K-step Markov Importance on the MAG (represented as a bidirected graph), weighted by normalized G^2 values, to refine the

feature set chosen with Program 6.1. This selects features by an importance measure that is related both to strength of association (and not only positive association) as well as movement through the graph of relationships. Therefore it will choose the most interesting nodes on paths up to k distance. Viewing feature selection as a network analysis problem allows strength of association, distances, and node degree to all be taken into account.

Our feature selection algorithm is as follows. The association rule mining phase will first choose the broad domain, such as pregnancy. Then the network analysis phase will refine the feature set, by choosing all features that follow specific situations, such as complications of pregnancy. We refer to such nodes as *non-actions*, which include diagnoses and complaints. Likewise, *actions* are all other CPOE orders. We desire the most important actions for the most important non-actions in the network. Therefore we run k-step Markov Importance twice, once each to select non-actions and actions. First, we set the root nodes to all the actions in a MAG generated on a dataset from Program 6.1. We run k-step Markov Importance and select the top n_1 non-actions. We then remove all non-selected non-actions from the MAG and rerun k-step Markov Importance on the modified MAG, this time with non-actions as the root nodes. We then choose the top n_0 actions. Selecting the size of n_0 and n_1 is fairly arbitrary and is best chosen by considering restrictions on computational performance in learning a Bayesian network on the selected feature set size. The program is described in Program 6.2, and a visual example is shown in Figure 6.4.

Program 6.2 runs in less than 30 seconds on all networks we have explored, including a MAG for the entire GopherInpatient dataset. It does fist require generation of the MAG, but we demonstrated in Chapter 5 that our SQL approach can learn a MAG in a sparse dataset very rapidly as well. Better, time spent learning the MAG is not lost. We will see in Chapter 7 that we can use the MAG to significantly speed up Bayesian structure learning. Even though a subset of features in the MAG will be chosen for the final graph, the MAG for a subset of features is the subset of the MAG for those features, because the original MAG contains all possible edges between variables. Therefore the same MAG can be used for learning after feature selection.

Program 6.2 Feature-Importance(D,V,n_0,n_1,k)

Input: D is a data set of transactions, consisting of items in a transaction
Input: V is the maximal set of possible variables, such as those output by Program 6.1
Input: n_0 is the maximum number of actions to choose
Input: n_1 is the maximum number of non-actions to choose
Input: k is the number of steps in k-step Markov Importance
  G = the MAG using D,V (Output of Program 5.1)
  for $i = 0 \rightarrow 1$ **do**
    if $i = 0$ **then**
      Root nodes are actions
    else
      Root nodes are non-actions
    end if
    ρ_0 is a vector of starting states with a uniform distribution among the root nodes
    A is a state-transition matrix representing the MAG weighted by normalized G^2 values
    $\rho_k = \sum_{t=0}^{k} A^t \rho_0$ {Perform k-step Markov importance.}
    S_i = the n_i features with the highest value in ρ_k which are not root nodes.
    $G = G \cap (S_i \cup \text{rootnodes})$ {Remove non-selected, non-root nodes from the graph.}
  end for
  return $\sum_i(S_i)$ {Return both selected actions and non-actions}

3. Evaluation

3.1. Implementation. We sought to compare our graph-theoretic algorithm to the frequency heuristic used in Chapters 3 and 4. Therefore we defined the following two programs:

- FS-IMPORTANT(D,T,n_0,n_1,n_{1p}): The full algorithm developed in this chapter, which runs an initial association-rule-mining pass (Program 6.1), generates the MAG (Program 5.1), and concludes with FEATURE-IMPORTANCE (Program 6.2). D is an initial dataset, T is a set of initial targets on which to perform association rule mining, and n_0 and n_1 are the number of actions and non-actions to select for the refined set, respectively. n_{1p} is an optional parameter that specifies the number of non-actions to *pursue* in the action-selection run of Markov importance. Only the top n_{1p} non-actions are set as targets in the second pass. This allows feature selection to focus on a subset of nonactions if the main targets for action selection are a subset of non-actions. (This is the case in our CHF evaluation below, where we are most interested in treating CHF, not complications thereof.) If n_{1p} is not specified, then $n_{1p} = n_1$.

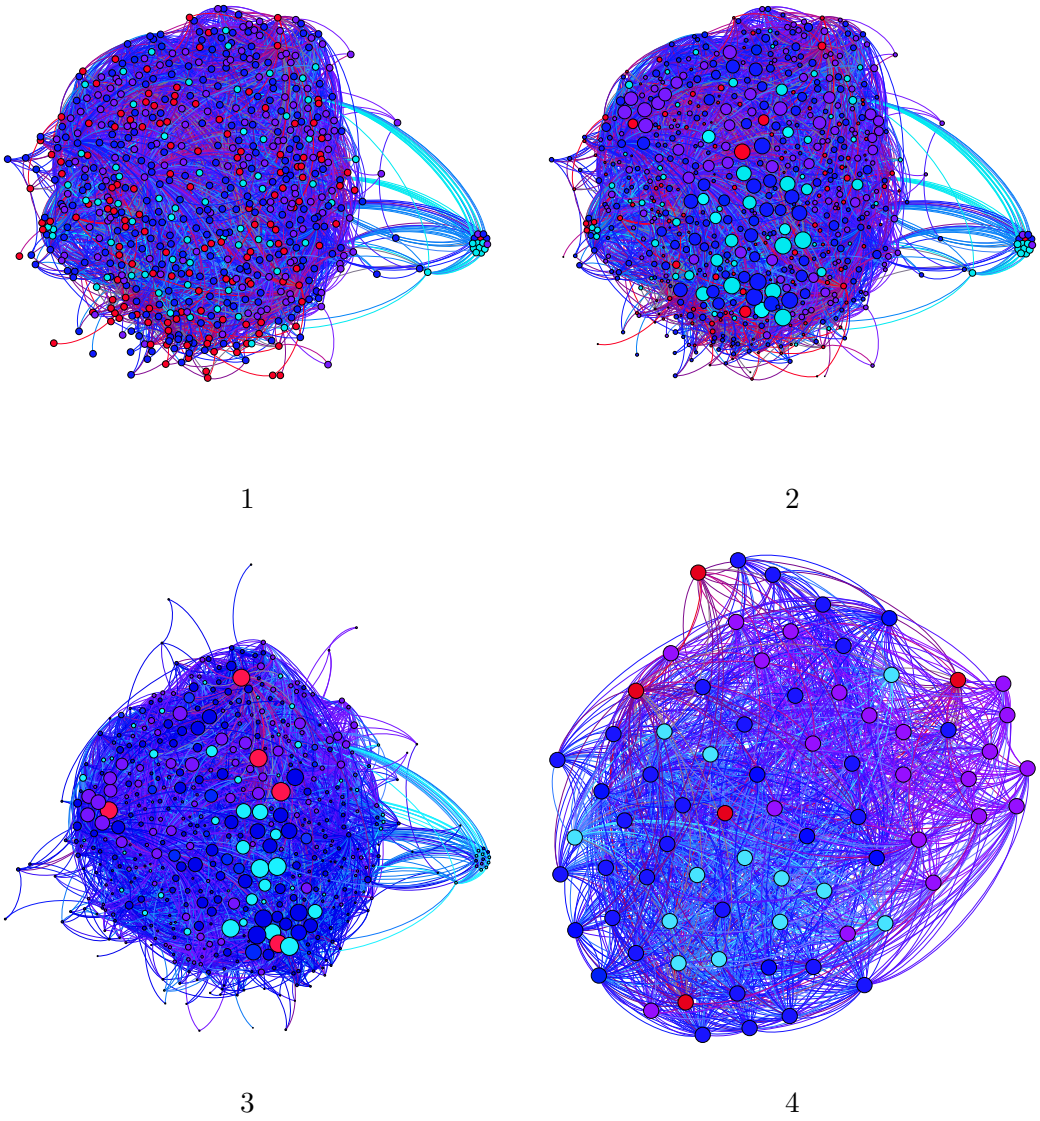


Figure 6.4. A visual example of Program 6.2, with colors as in Figure 5.5. (1) The MAG generated on the GopherInpatient dataset after performing feature selection (Program 6.1) with T={Pregnancy}. (2) The nodes in the MAG are sized by their rank with actions as targets. (3) Non-selected non-actions are removed and the nodes in the MAG are sized by their rank with remaining non-actions as targets. (4) The final graph after all non-selected nodes are removed.

- FS-FREQUENT(D,T,n): The frequency heuristic from Part 1, which counts the co-occurrences of a set of targets (T) on a dataset (D). The most frequent n features will be selected.

Our implementation of FS-IMPORTANT was as follows. First, we implemented Program 6.1 as a SQL script. (This is actually ChoosePairs and FilterPairs from Klann et al. (2009), but modified to compute only confidence.) We then wrote a SQL script to generate temporary data tables using the subset of variables selected by Program 6.1, on which we generate the MAG using Program 5.1. Next, we developed a converter to load the edgelist output by the MAG generator into a JUNG graph, which is OMadadhain et al.'s (2005) Java graph framework. Because JUNG already provides an implementation of k-step Markov Importance, we were able to implement the two-step importance filtering in Program 6.2 directly using JUNG's application programming interface (API). Finally, we wrote a Java program to execute these various pieces (assocation rule mining, MAG generation, JUNG conversion, and importance sampling) automatically given a source dataset and a set of initial targets.

We chose our parameters t and k as follows. After experimentation with the Gopher-Inpatient dataset and $T = \{Pregnancy\}$, we found $t = .01$ reduced the feature set by 80% without eliminating relevant features. White and Smyth (2003) suggest a $k \leq 10$. We reasoned that too large a k would focus selection too heavily on orders which are co-associated rather than associated with target diagnoses. Therefore we chose an approximation of treewidth as our k, up to a maximum of 5 for non-action selection and 3 for action selection. Recall that treewidth is the width of the optimal junction tree decomposition of a graph and is related to the number of paths between nodes (and therefore the number of colliding paths). Determining treewidth exactly is NP-hard (Lucena, 2003), but good approximations exist. We defer discussion of these approximations to Chapter 7, and for the moment simply assume we can approximate treewidth through a program TW(G). Then $k = \max(5, TW(G))$.

Our implementation of FS-FREQUENT simply runs Program 6.1 to compute co-occurring variables and then selects the first n without computing confidence. FS-FREQUENT ran somewhat faster than FS-IMPORTANT but the differential was always under 5 minutes.

3.2. Methods. We hypothesized that the features chosen by FS-IMPORTANT are superior to FS-FREQUENT. To test this hypothesis, we developed the following two measures:

- **Length for key features.** We measured the length of a list that must be chosen to select the essential basic features for a domain using both programs.

- **Feature promotion and demotion.** We call the features selected earlier by by FS-IMPORTANT than by FS-FREQUENT *promoted*, and those selected later *demoted*. This is a measure of how important FS-IMPORTANT considers the feature compared to what frequency would suggest. We examined the nature of promoted and demoted features.

We applied our measures to two domains in the GopherInpatient dataset, one involving Congestive Heart Failure (CHF) and one involving pregnancy. The former is a fairly small domain and straightforward hospitalization, whereas the latter (as we have seen), is frequently wrought with complications.

For the CHF evaluation, we chose to compare the most important treatments for just that single target (CHF), so we ran the metrics on GopherInpatient with $T = \{\text{CHF}\}$ (and $n_1 = 5$,$n_{1p} = 1$ for FS-IMPORTANT). To compare list length for key features, we selected the four basic treatments considered in CHF hospitalizations according to a Gopher order set: an ACE inhibitor, cardiac glycoside, diuretic, and nitrate. To study feature promotion and demotion, we compared the first 200 actions chosen by both programs.

For the pregnancy evaluation, we ran the metrics on GopherInpatient with $T = \{\text{Pregnancy}\}$ and $n_1 = 5$ for FS-IMPORTANT. To compare list length, we selected features chosen as the key treatment by the obstetric nurse in Chapter 3 for the non-actions chosen by FS-IMPORTANT. To study promotion and demotion, we compared both the first 200 and top 5 non-actions chosen by both programs.

3.3. Results. We report the list length required to choose key features for both domains in Table 6.1. As mentioned, in the CHF domain the key features were the four essential treatments in a Gopher order set. In the pregnancy domain, they were the key treatment indicated by the obstetric nurse (see Chapter 3) for the three comorbid conditions selected by FS-IMPORTANT also evaluated in Chapter 3 (two of the non-actions chosen by FS-IMPORTANT were not evaluated in Chapter 3, so we only consider three features here).

We report feature promotion and demotion for pregnancy complications in Table 6.2. The different choices for non-actions is shown, as well as a representative sample of treatments with high promotion and demotion. We report feature promotion and demotion

for CHF in Figure 6.5. A representative sample of treatments with high promotion and demotion is shown alongside a histogram.

Evaluation	Treatment	FS-Important	FS-Frequent
Pregnancy	Oxytocin	17	12
	Magnesium Sulfate	18	65
	Betamethasone	7	80
	Total length	**18**	**80**
CHF	ACE Inhibitor	84	11
	Cardiac Glycoside	23	158
	Diuretic	1	30
	Nitrate	14	19
	Total length	**84**	**158**

Table 6.1. Comparison of the list length at which key features are selected. The list length to capture all common relevant features is shown at the bottom of each section.

FS-Important	FS-Frequent
Postpartum	Postpartum
Cesarean Section	Cesarean Section
Preterm Labor	Spontaneous Vaginal Delivery
Preeclampsia	Early Stage Labor
Premature Rupture of Membrane	Tubal Ligation

(a) Top pregnancy problems by program.

Treatment	Promotion
Transvaginal Ultrasound	94
Fetal Ultrasound	75
Nitrofurantoin	59
Vitals	-54
Regular Diet	-55
Tylenol w/ Codeine	-66

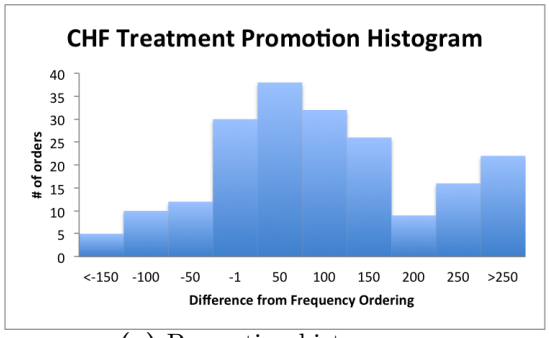
(b) Representative examples.

Table 6.2. FS-Important compared to FS-Frequent for inpatient pregnancy. (Left) The top five cormorbid problems in inpatient pregnancy, FS-Important compared to FS-Frequent. (Right) A representative sample of treatments with high differential in ranking. Features with high promotion are highly relevant in specific situations. High demotion features are extremely general.

3.4. Discussion. FS-Important consistently outperformed FS-Frequent.

In the pregnancy domain, Table 6.1 shows the three key features were chosen with a set of size 18, rather than 80 with FS-Frequent. Table 6.2 shows FS-Important's top five comorbid diagnoses promoted complications, displacing a routine procedure and two diagnoses that merely provided more information (e.g., *early stage labor*). Finally, the most promoted features were frequently quite important among the top five comorbid diagnoses and were frequently found in the nurse's lists in Chapter 3, whereas general treatments (e.g., strong painkillers) were heavily demoted.

In the CHF domain, FS-Important chose essential basic features in half the number of features as FS-Frequent (Table 6.1). It performed best with the rarest essential feature (cardiac glycoside) and worst with the most common (ACE inhibitor). FS-Important demoted nearly all non-specific features significantly (e.g., multivitamin and ranitidine), and filled in the list with less frequent treatments that could become essential in some cases. (See Figure 6.5.) Most strikingly, a cluster of four ventilator protocol orders were



(a) Promotion histogram.

Order	Promotion
CHF Education	325
Ventilator Protocol	229
Restrain	211
Resp. Therapy Consult	166
Anticoagulation Clinic	122
Vancomycin	107
Spironolactone	95
Oxygen Therapy	28
Cardiac Markers	-26
Multivitamins	-117
Ranitidine	-308
Sudafed	-311
Promethazine	-333
Acetaminophen	-392
Percocet	-472
Ambien	-744

(b) Representative examples.

Figure 6.5. (Left.) A histogram of the difference in feature selection order in FS-Important compared to FS-Frequent, for the top 200 features chosen by FS-Important. (Right) A representative sample of orders with high differential among the top 200 feature chosen by each program. Features with high promotion are highly relevant in specific situations. High demotion features are extremely general. The middle tier of both promotion and demotion are relevant in many situations.

added, which are more 'important' than a multivitamin in that they are more specialized, even though less than 10% of CHF hospitalizations involve a ventilator at Wishard.

The only situation in which FS-Important performed poorly was in choosing an ACE inhibitor for CHF. This is related to how the algorithm views the domain. In FS-Important, feature choice is related to the strength of paths from roots to treatment. Because our CHF evaluation involved only one root, overall frequency in the domain can occasionally be a stronger predictor of relevance than importance. FS-Important excels at capturing complexity in a domain of multiple problems and evolving treatments.

Overall, FS-Important selects a more relevant list of features in a smaller set size, for both complex and simple domains. Also, within that list length, FS-Important chooses much better remaining features, selecting less common but highly relevant features over non-specific common ones.

3.4.1. *Limitations.* As discussed, FS-Important is a heuristic that does not always select the optimal feature set. The algorithm's primary weakness is that the edge weight, pairwise G^2, is only an approximation of maximal association strength. In particular, the G^2 value can vary as the conditioning set changes, so the only way to know the maximum G^2 value is to test the node conditioned on subsets of all neighbors of both nodes. This, like other algorithms mentioned at the beginning of this chapter, has the same time complexity as learning the Bayesian network and is therefore NP-hard.

We argue that in most cases, an unconditioned G^2 value is 'good enough', for two reasons. First, the more paths between nodes, the more likely some G^2-weighted path will reflect the true importance of the feature. Because our feature selection graphs have thousands of features and we have seen in most domains that certain 'hub' nodes are highly connected, multiple paths will frequently exist between nodes. Further, we expect dramatic differences by conditioning G^2 will not occur in the real world. Although it is possible to construct endless theoretical examples in which two nodes that are almost uncorrelated become very strongly associated by a third (see Figure 6.6), it is difficult to find a such a dramatic example in the domain of medical treatment. Even our motivational example for considering multivariate networks, the asthma order set example from Chapter 2, a snippet of which is again shown in Figure 6.6, is not impacted by this. Although terbutaline

should only be considered for those under 40, the unconditioned association between asthma and terbutaline will still be relatively high, because many patients are under 40 (and so conditioning on age does not dramatically shift the association). That is not to say that variables do not become more correlated in the presence of other factors - in fact the ability to model such changes is a key strength of Bayesian networks - but only that the shift is not so dramatic that feature selection will frequently fail.

An inaccurate path of G^2 values only matters when it is the only path between a target and a node. Because we are selecting among thousands of features (meaning the graph has many paths) and because we have found orders in medical subdomains are frequently highly connected, we expect multiple paths to lead to most nodes. Therefore only extremely rare and specific orders could suffer this fate.

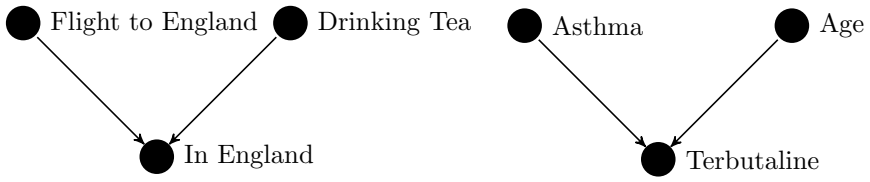


Figure 6.6. Hypothetical examples of conditioning variables' impact on posterior probability. (Left) Correlation varies dramatically when a conditioning variable is introduced (tea becomes highly correlated with being in England only after a recent flight there). (Right) A more realistic example in which the conditioning variable has a moderate impact on correlation (terbutaline should only be prescribed to asthma patients under 40).

Nonetheless, developing an association network that more closely approximates maximal G^2 values is an important open research problem.

3.4.2. *Conclusion.* FS-IMPORTANT chooses an overall set of much more relevant features in all domains, where the differential as compared to a frequency heuristic is more significant in more complex domains. It does as well as a frequency heuristic at choosing the essential basic features of simple domains.

CHAPTER 7

Scalable learning in sparse domains

Although feature selection will allow us to select domain-specific networks, the complexity of medicine would still suggest we need a Bayesian structure learning algorithm that scales well. For example, even if we only choose the 200 most relevant features in our order-entry data for a domain, once we include some metadata (such as demographics and relevant test results) and involve temporal reasoning (many methods for which, as we shall see, minimally double the number of time-varying features), we have a network of several hundred features. Further, if we consider the by-session Gopher dataset from Chapter 1 (rather than the admission-compressed dataset), we will likely have tens of thousands of samples.

The GES algorithm used in Part 1, along with all other score-based network learning algorithms, perform poorly in such large networks. There are other classes of algorithms designed for *sparse* domains, in which the number of edges is a relatively small percentage of possible edges. Nevertheless, even these algorithms do not perform as well as we might like. As mentioned in Chapter 5, the Max-Min Hill Climbing (MMHC) algorithm (which is optimized for sparse domains) took 13 days to learn a network for a 5000 variable by 5000 sample dataset on a 2.4GHz Pentium Xeon (Tsamardinos et al., 2006). Our networks will have many fewer variables but potentially many more samples. It is not unlikely that MMHC would take a day to learn such a network, and we would like to learn many such networks. This is not conducive to experimentation and exploration, or even a system running in a rapidly changing environment.

Therefore in this chapter, we will develop a Bayesian structure learning algorithm optimized for our application: a sparse domain of predominantly binary variables with a moderate amount of data. We will then demonstrate that its speed outperforms any existing algorithm yet yields comparable predictive performance. First, we will review Bayesian structure learning algorithms designed for sparse domains.

1. Review of Approaches

1.1. Terminology. It is helpful to establish a few definitions related to network learning.

DEFINITION 2. *A graph G is* FAITHFUL *to a dataset D if and only if every conditional independence assumption present in the graph is also present in the data (Tsamardinos et al., 2003). Moreover, an algorithm is* SOUND *if it always finds a graph faithful to a dataset, if one exists.*

In inconsistent or noisy data, a graph faithful to the data cannot always be found, but nevertheless in practice many algorithms can find a 'good enough' graph. How the algorithm reacts to noisy data is called the *stability* of the algorithm.

DEFINITION 3. *An algorithm is* STABLE *if it is able to discover the predominant patterns in a noisy or inconsistent dataset.*

Multiple graphs exist that are faithful to the data. These are called INDEPENDENCE MAPS (I-maps). I-maps can have useless edges that do not reflect dependencies in the data (one such I-map is the fully-connected graph). Therefore a MINIMAL I-MAP is preferred; this is a graph in which removing any edge violates the faithfulness condition. There might be multiple minimal I-maps, the smallest possible of which is known as the PERFECT MAP (or P-map) of the distribution. Therefore one view of the goal of Bayesian network structure learning is to find a minimal I-map that is as small as possible (Koller and Friedman, 2009).

We use I-map and P-map loosely, using them to refer to both directed graphs and graph skeletons. This is done to simplify the discussion, but I-maps are actually *partially directed graphs* that have all undirected edges except for colliders. Such graphs determine an *equivalence class* of graphs in which all orientations are probabilistically equivalent as long as no additional colliders are introduced. (This is the search space of the GES algorithm.)

DEFINITION 4. *A minimal I-map is a graph G faithful to a dataset D in which removing any edge violates the faithfulness condition. A* P-MAP *is the smallest minimal I-map.*

In Chapter 2, we discussed greedy searches, and we said the greedy equivalence search (GES) algorithm is provably the best-performing score-based searcher. Now we can be more precise. All greedy search algorithms are both sound and find a minimal I-map, but they are not in general guaranteed to find the P-map, which is why random restarts are employed. GES is, in contrast, guaranteed to find the P-map (if one exists). A strong

feature of score-based searches are that they are reasonably stable. However, they do not perform well in sparse domains.

1.2. Bayesian Network Learning in Sparse Domains. The predominant alternative to score-based learning is *constraint based learning.* Such algorithms, extensively described in Spirtes et al. (2000), utilize a series of conditional independence tests to determine D-separation and discover the graph skeleton. (Then further tests are performed to orient the graph.) Because they eliminate potentially edges through tests for local structure (rather than global searches through an unconstrained edge-space), they are faster in sparse domains.

As alluded to in Chapter 5, the simplest constraint-based learning algorithm would reconstruct a graph skeleton by performing a D-separation test on every pair of variables, conditional on every subset of every other variable in the dataset. Then, if no conditional set D-separates the pair, there is an edge between them. This sequence of steps is used in the SGS algorithm. SGS is a sound algorithm and it is quite stable up to the point of finding the skeleton. Orienting the graph, done using information on D-separation subsets, is less stable. However, given consistent data, SGS (and most other constraint-based algorithms) will find the P-map.

It is important to note that SGS' guarantees break down when the statistical test produces incorrect results. Practically, this can be a problem because the algorithm suffers from the multiple hypotheses problem. In particular, if each conditional independence test has a confidence level of 95%, there is a 5% chance for every test that the null hypothesis is rejected when it should not be. This leads to false positive associations between variables, which in large graphs can accumulate to many errors. Likewise, because tests are performed on the same variables conditioned on multiple subsets, false negatives can accumulate as well. In statistics, an adjustment of the confidence level is usually performed in the case of the multiple test problem, such as the Bonferroni correction. However, there is no proven method to choose an optimal correction value for complex algorithms like SGS. This concern is mitigated somewht by *hybrid algorithms* that combine constaint-based and score-based learning. We will come to these algorithms shortly, but in many cases, SGS-style algorithms find graphs that are at least as good as many score-based algorithms.

This section is about algorithms for sparse domains, but SGS does not perform well on sparse domains. In fact, it does not perform well in *any* domain, as it requires $O(n^{n+2})$ conditional independence tests. However, the insight of SGS is that one can use D-separation checks to determine edges. Therefore only local tests need to be performed (between pairs of variables) rather than global scoring checks (on all possible edges).

Spirtes and Glymour (1991) developed SGS primarily for its theoretical properties. They apply these properties in an tractable algorithm called PC. Beginning with the fully-connected graph, it progressively 'thins out the edges' through D-separation tests on increasingly large subsets of variables. In this way, many erroneous edges are eliminated in the first several passes, and each pass takes only $O(n^{1+p})$ operations, where p is the pass number. In the worst case, PC is as poor as SGS, but PC takes advantage of domain sparseness, so often only a half-dozen passes are needed.

The fastest sound constraint-based algorithm is most likely Pellet and Elisseeff's (2008) Collider Sets (CS), which on average performs an order of magnitude fewer calculations than PC and takes on average $O(n^4)$ operations, though it still has exponential running time in the worst case.

Friedman et al. (1999) arrived at much the same conclusion about sparse domains - that potential edges can be limited through local tests - with a different approach. They introduced a class of algorithms known as Sparse Candidate, which perform a score-based search but first remove edges from consideration that are not likely to be connected. Friedman did this by, for each node, choosing the nodes with the top ten or fifteen largest mutual information as possible parents. Friedman found that beginning each restart of a greedy search with a mutual-information-based edge-reduction algorithm (based on the results of the previous pass), the resultant searches converged much faster than greedy search alone. He found that learning a network of 100 variables took about half the time and half the number of statistical calculations as a standard greedy search to reach convergence. With 200 variables, the speedup was more than three. His expectation is that this approach will continue to show similar gains as the network size grows, so long as the network is sparsely connected.

A newer approach, inspired by both Sparse Candidate and constraint-based algorithms, are *hybrid algorithms*. These use a constraint-based algorithm to find the graph skeleton (or some superset of it) and then invoke a greedy search to orient the graph. This borrows from the best of both worlds. In sparse domains, the speed of D-separation to find the skeleton is unparalleled. However, orienting the graph through D-separation is, as discussed, unstable, whereas score-based algorithms orient based on a global view rather than local tests. Moreover, Tsamardinos et al. (2006) found that hybrid algorithms which generate a faithful undirected skeleton produce very accurate networks (when compared with a graph used to generate the data) with only a one-pass greedy search (no restarts). There is a time tradeoff for learning the true undirected skeleton rather than a simpler heuristic, but overall the authors found their Max-Min Hill-Climbing Algorithm (MMHC) was 15 times faster than Friedman's 15-candidate mutual-information algorithm with a sample size of 5000. (Incidentally, it was also 167 times faster than an unoptimized version of GES.) Its average case complexity is better than PC: $O(n^2|PC|^{J+1})$, where n is the number of nodes, $|PC|$ is maximum number of neighbors of a node in the skeleton, and J is a user-specified constant related to the maximum treewidth expected in the graph, which is typically small.

1.3. Efficient Skeleton Discovery. All of the foremost approaches to learning in sparse domains begin with discovering the undirected skeleton. We discussed the approach of PC, and now we will explore a family of faster approaches used by algorithms like MMHC. These involve Markov blankets, which, as discussed in chapter 6, is the set of variables most relevant in prediction. In Bayesian networks, it also has a graphical meaning: it is the set of parents, children, and siblings (other parents of children) of a node [1]. In cases where the data is faithful to some network, the Markov blanket is unique (Margaritis and Thrun, 1999). Thus to discover the Markov blanket is to discover the P-map surrounding any particular node of interest.

This set of all Markov blankets can be 'stitched together' to form what is *almost* the skeleton of the graph. The missing piece is that Markov blankets include siblings, so stitching all of the blankets results in what is called the *moral graph*. Several Markov Blanket

[1]Actually this is the Markov Boundary, which is the minimal Markov blanket. Here we will refer to this as the Markov blanket.

discovery algorithms (e.g., Max-Min Markov Blanket) lose the siblings during their search for parents and children and have to explicitly find them again. In this case, the sibling-less blankets can be stitched together directly.

Another way to strip siblings from a Markov blanket is through a triangle rule. This is alluded to by Pellet and Elisseeff (2008), which we formalize here. Siblings in the Markov blanket are always part of *triangles*, as illustrated in Figure 7.1. We make an observation about such triangles.

DEFINITION 5. *Say there is (triangle A-B-C) in the moral graph. Then if any two nodes in the triangle are D-separated by the third node, the two nodes are siblings with a common child.*

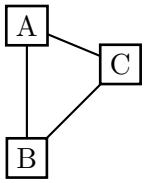
Definition 5 arises from our discussion on D-separation and intuition developed in Figure 7.1. We can therefore find all siblings in the moral graph by checking for D-separation on every pair of nodes conditioned on common neighbors. This involves a D-separation test for every node in each Markov blanket conditioned on every disjoint subset of size 1 in that Markov blanket. This takes $O(|MB|^2)$ operations, where $|MB|$ is the size of the Markov Blanket. To our knowledge, this triangle rule is only used by Pellet and Elisseeff's (2008) CS, to find potential common parents. Other Markov-blanket skeleton discovery methods utilize algorithms that generate sibling-less Markov blankets.

Having established a method to reconstruct a network skeleton from any Markov blanket discovery algorithm, we review Markov blanket discovery algorithms.

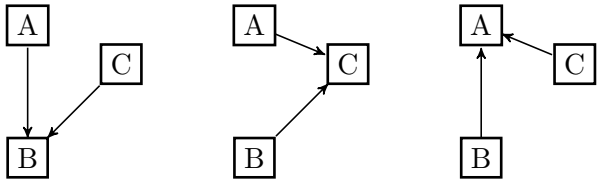
Margaritis and Thrun's (1999) grow-shrink (GS) algorithm is the fastest, operating in time linear to the number of samples. It uses conditional independence tests and the principle that the Markov blanket uniquely D-separates a target variable from all other variables. It steps through the variables in the data, adding variables to the blanket through successive conditional independence tests on the current blanket. Then a *shrink* phase follows, which removes variables erroneously added to the blanket, via a series of loops over the discovered Markov Blanket. GS is sketched out in Program 7.1. The disadvantage is that it requires on average a sample size exponential to the size of the Markov blanket. In the worst case, the available samples must be exponential in the number of variables in the

Node	Markov Blanket
A	B,C
B	A,C
C	A,B

(a) Markov Blankets



(b) Moral Graph



(c) Orientations After Sibling Edge Detection

Figure 7.1. The structure of a sibling relationship discovered through connecting Markov Blankets. (a) shows the Markov Blankets of several nodes and (b) shows their corresponding triangle structure in the moral graph. (c) shows possible orientations of the triangle with the sibling edge removed. These sibling edges can be discovered by checking for D-separation on each pair in the triangle, conditioned on the third variable.

data, because erroneous variables might be admitted to the blanket and are only removed at the end, which can make the intermediate blanket size much larger than the actual blanket (Tsamardinos et al., 2003). GS is provably correct given a sample size exponential in the number of variables in the data. All other work on Markov blanket discovery has been to reduce the sample limit.

Program 7.1 Grow-Shrink(T,D)

Input: T is a target variable for which to find the Markov Blanket
Input: D is a a training dataset
  MB is an array of variables in the Markov Blanket, initially empty
  for all Variables $v \in D \neq T$ **do**
    if not $v \perp T|MB$ **then**
      $MB \leftarrow v$ {Add v to the blanket.}
    end if
  end for
  repeat
    for all Variables $v \in MB$ **do**
      if $\not\perp T|MB$ **then**
        $MB \setminus v$ {Remove v from the blanket.}
      end if
    end for
  until the shrink loop does not change the blanket.
  Return the MB

Tsamardinos et al.'s (2003) iterative-associative Markov blanket (IAMB) algorithm chooses variables in a better ordering when admitting them to the Markov blanket. Rather than checking variables in arbitrary order, it finds the most-associated variable with the current Markov blanket. In other words, it finds a variable v that maximizes $G^2(T; v|MB)$, where $|MB|$ is the current draft of the Markov blanket and T is the target variable[2]. This involves checking all remaining variables in every pass through the grow loop, so IAMB takes on average $O(|MB|n)$ statistical tests (worst case $O(n^2)$). The advantage is this method reduces the false positives that enter the blanket, so it requires a sample size closer to exponential in the size of the largest Markov blanket. It also is quite stable, whereas GS is not, making it a good choice for noisy datasets.

Aliferis et al.'s (2003) HITON algorithm (and its cousin, Max-Min Markov Blanket) reduces the required sample size further by testing every candidate variable on all subsets of the current blanket (up to some fixed size) before admitting it to the blanket. (As discussed, this severs siblings, which must be re-discovered later if the Markov blanket and not the skeleton is desired.) It does this by augmenting the method in IAMB, adding a loop after a candidate is chosen to perform subset checking. The result is that D-separation between the target and a candidate is found early, well before the entire blanket is discovered, thereby eliminating many false positives. Thus HITON requires a sample size exponential only in the size of some small constant k — the maximum size of the conditioning set. Of course, it is also much slower than either of the preceding two algorithms, because it requires up to $\wp(k)$ tests on every candidate variable. In the average case, HITON runs in $O(|MB|^2 n)$ time, but the worst case running time is now exponential in k. In actual time, one experiment involving 139,000 possible variables found the Markov blanket in approximately 8 hours using an Intel Xeon 2.4 GHz computer with 2 GB RAM. Tsamardinos et al.'s (2006) Max-Min Hill-Climbing algorithm, uses a method similar to HITON to find the parents and children of each node, which it then arranges into the undirected skeleton.

Finally, we note there are heuristic algorithms that have no guarantee of soundness (most famously Koller and Sahami's (1996) KS algorithm), but they are slower and have frequently poor results, though they do not require exponential sample size.

[2] G^2 can be replaced by any similar statistical test, such as χ^2.

1.4. Skeleton Orientation. Once a network skeleton is discovered, the final step is orientation. Constraint-based learning algorithms, such as Pellet and Elisseeff's (2008) Collider Sets (CS) uses additional local statistical tests to orient the graph, but in the limit these can be quite slow and they are also inherently unstable. As discussed, other sparse-domain algorithms conclude with a score-based greedy search, which is much more stable. Tsamardinos et al. (2006) found that restarts are not necessary on a faithful skeleton, making this orientation process quite fast.

2. Fast hybrid learning algorithms

There is significant room for speed improvements over MMHC if we are willing to make some sacrifices.

2.1. Maximal Association Graphs in hybrid learning: MAG-GS. If we sacrifice the guarantee that the algorithm will find the P-map, we could orient an approximate skeleton. As long as the approximate skeleton is not missing any true edges, the algorithm will still find a minimal I-map. The closer the approximate skeleton is to the correct one, the better minimal I-map that will be found. Perrier et al. (2008) call such an approximate skeleton a *super-structure*.

Definition 6. *A super-structure is any undirected graph that contains the actual graph skeleton (Perrier et al., 2008).*

One approximate skeleton is the Maximal Association Graph (MAG), developed in chapter 5. Recall that the MAG uses D-separation to find a set of all edges that *might* exist in the final graph. This requires $O(n^2)$ operations in an iterative implementation, but in Chapter 5 we created a version for binary variables that runs on relational databases extremely rapidly in sparse domains. It also does not have any notable sample size requirements. Contingency tables in the MAG always have four entries (see 5.2), requiring only four partitions and therefore a sample size of < 100. Therefore the sample size requirement is bounded by the greedy search and not the skeleton finding.

Therefore we introduce a hybrid algorithm consisting of the MAG oriented by a greedy search, which we call MAG-GS. The more complex the graph, the less likely it is to find the P-map, but its stability and speed make it an appealing hybrid algorithm. This method

is shown in Program 7.2, which relies on the greedy search in Program 3.1. We also make one innovative addition to greedy search that requires more discussion.

Program 7.2 MAG-GS(D)

Input: D is a a training dataset of binary variables
MAG ← Run Program 5.1 on D. {Learn the MAG.}
Run REDUCE-TW(GREEDY-SEARCH(D,R),D), where R is the MAG.

2.1.1. *Bounding Tree-Width.* As discussed in chapter 5, complexity of inference is highly related to the tree-width of the graph. Unfortunately, finding the tree-width of graphs is NP-hard, but there are many upper- and lower- bounds that can be computed rapidly. We would like to use tree-width to aid our greedy search. Although scoring functions already penalize graph complexity, a graph of the same complexity with a higher tree-width is, for the purpose of inference, a 'worse' graph. Limiting tree-width is especially important in an algorithm that does not guarantee a P-map, because minimal I-maps differ in complexity and thus might have different tree-widths.

Work has been done by Elidan and Gould (2008) in monitoring the tree-width of a graph as it is being learned, which can guide the search toward I-maps with smaller treewidth. There is a significant speed loss for making these changes, however, because additional calculations must be performed for every edge addition to judge its potential contribution to tree-width. Such methods are likely very important on large graphs when a greedy search is unconstrained, but are less necessary when beginning from a sound skeleton. This is because such algorithms get close to the P-map already, so online guidance toward better I-maps is not necessary.

Therefore we use this heuristic: once the graph is learned, if the (approximate) treewidth is above a threshold max-treewidth, we delete edges that reduce the score the least until the (approximate) treewidth is max-treewidth. Of course, if edges are deleted and the output graph was in fact a minimal I-map, then the graph after treewidth reduction is no longer faithful to the dataset. Therefore we must use this optimization cautiously. In practice, however, score-based learners tend to add a few extra edges, and the sensitivity in inference to minor problems in BN specification is frequently quite low (Pradhan et al., 1996). Only when many edges are deleted is this practically problematic (e.g., running

after an unconstrained greedy search). In general, the more optimal the network skeleton, the fewer edges will be deleted by this heuristic. We will introduce the BS and MAG-BS algorithms in the upcoming sections to find better skeletons.

To compute approximate tree-width, we use the Greedy Degree metric, which van Dijk et al. (2006) found to be an upper bound which was frequently very close to exact tree-width. It can be computed in time linear to the number of nodes, and because it only requires graph manipulation (rather than expensive statistical tests), it is extremely fast ($< 1s$ on graphs we have tested). It works by finding the node of lowest degree and recursively removing this node (and its edges) until the graph is empty. The maximum degree among the lowest-degree nodes approximates the tree-width. This program is described in Program 7.3, and an example run can be found in Figure 7.2. Program 7.4 is a simple application of TW to remove low-scoring edges until a particular maximum tree-width is reached.

Program 7.3 TW(G)

Input: G is a graph for which an upper bound of tree-width will be calculated.
TW is the tree-width of the graph, initially 0.
while G is not empty **do**
  n = the node in the graph of lowest degree
  if degree(n)>TW **then**
    TW=degree(n)
  end if
  $G = G \backslash n$ {Remove n and its edges from the graph}
end while
return TW

Program 7.4 Reduce-TW(G,D)

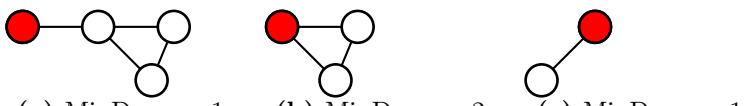
Input: G is a learned graph.
Input: D is a a training dataset.
while $TW(G) >$ *max-treewidth* **do**
  Score all edge removals in G, and remove the edge that decreases the score the least.
end while
Return G

2.2. Faster skeleton discovery: Blanket-Stitch. If, like MMHC, we would like to find the P-map, but we are willing to sacrifice the small bounds on sample size, we could replace the undirected skeleton phase of MMHC with IAMB (augmented by our triangle

rule). This would reduce the number of conditional independence tests for skeleton finding from $O(n^2|PC|^{J+1})$ to $O(n^2|PC|^2)$ in the average case.

When available sample size is extremely limited, MMHC's very small sample size requirements are important (such as structure discovery in protein interaction networks). However, when a Bayesian network is being learned, a larger sample size must be available regardless. Recall from Chapter 5 that Bayesian networks require a sample size exponential in the maximum number of parents.

For learning network structure for Bayesian networks, we introduce a new algorithm: Blanket-Stitch (BS). BS' Markov-blanket learning is as follows. While the candidate Markov Blanket (CMB) is smaller than $log_2(S/20)$ (where S is the sample size), we perform an IAMB-style search (because it is fast and more stable than Grow-Shrink) with one modification. We use the triangle rule presented earlier to check each candidate to verify it is not a sibling of any node in the CMB. When the CMB reaches the sample limit (or all potential neighbors have been considered), a Shrink phase is invoked which is similar to IAMB's except it also checks the CMB for siblings which might have been erroneously added. If the CMB is now below the sample limit, the algorithm continues. If it is still at the limit, we use a heuristic. Because (in most cases) the most-associated variables are added first, the most recently added variable is least likely to remove another variable from the CMB. Therefore we remove it from the CMB, add it to an overflow list (OMB), and continue the search. When the search completes, OMB∪CMB is returned as the Markov blanket. This will learn a Markov blanket in $O(n^2\|CMB\|)$ tests, where $\|CMB\|$ is the maximum blanket size allowed by the available samples. Although the OMB heuristic will occasionally admit false positives into the blanket, it is actually quite rare for the final skeleton to have many false positives. First, in graphs with moderate sample size and not many 'hubs,' the frequency of use of OMB is low. More importantly, false positives added



(a) MinDegree=1 **(b)** MinDegree=2 **(c)** MinDegree=1

Figure 7.2. An example run of Program 7.3. The treewidth upper bound is max(MinDegree)=2.

by the OMB are likely to be later removed by the skeleton generation process, because we allow each Markov blanket learned by BS to remove edges erroneously added to the graph by learning a previous Markov blanket.

BS stitches together blankets as follows. Starting with empty graph, each variable's sibling-less Markov blanket is iteratively added to the graph. Each variable's blanket is initialized with its preexisting neighbors in the graph, but those edges are later removed from the graph if the shrink phase removes them. For computational speed, we borrow an optimization from MMHC - only variables that have not been learned are considered for adding to the blanket (because if they were learned and are potential neighbors, they would already be in the graph).

Finally BS orients the graph using the greedy search already presented in Program 3.1 and limits the treewidth of the resulting graph through Program 7.4. This is described in Program 7.5. To keep the treewidth small (so Program 7.4 deletes at most a few edges), we limit the maximum fan-in for the greedy search to one-third of the maximum number of parents allowed by dataset size (see Chapter 5).

2.3. Combining Blanket Stitch and the Maximal Association Graph. We introduce our final hybrid algorithm. By limiting BS to search only among edges in the MAG, we increase its speed dramatically without sacrificing its ability to find the P-map. We call this algorithm MAG-BS. The skeleton-finding runtime of MAG-BS is $O(n^2 e \|CMB\|)$, where e is the maximum number of edges in the MAG. In chapter 5, we saw that the MAG for GopherInpatient contained less than 1% of all possible edges. We expect that at some number of edges the skeleton finding runtime will be the upper bound for learning the network, as the greedy orientation phase takes less time the more the edges are constrained.

3. Implementation and Evaluation

3.1. Methods. We implemented MAG-GS,BS, MAG-BS, and an unconstrained greedy search GREEDY (see Programs 7.5, 7.2, and 3.1) in Java, utilizing components in the Tetrad IV suite (Ramsey, 2011) including: the conditional G^2 test (in Program 7.5), a BDeu scorer (in Program 3.1), and various graph manipulation utilities (such as cycle checking).

Program 7.5 Blanket-Stitch(D,R)

Input: D is a a training dataset
Input: R is a restriction graph. Only edges in this graph will be considered. The graph is undirected.
G is an output graph, which initially is the empty graph.
L is a list of variables that have been learned, initially empty.
for all Variables $v \in D$ **do**
CPC = $\{\forall c \in G : c \sim v\}$ (We will refer to this as CPC_0 later.)
CONSIDER = $\{\forall c \sim v \in R : c \notin L \wedge c \notin CPC\}$
OPC = $\varnothing$
while $\|CONSIDER\| > 0$ **do**
CONSIDER $\bar{=} \forall c \in CONSIDER : c \perp v | CPC$
CONSIDER $\bar{=} \forall c \in CONSIDER \forall pc \in CPC : c \perp v | pc$
Find the variable $c_{best} \in CONSIDER$ that maximizes $G^2(c; v|CPC)$
{Find the most associated variable given the current CPC that is not a sibling according to the current CPC.}
CPC $\leftarrow c_{best}$ {Admit the best candidate into the CPC.}
CONSIDER $\bar{=} c_{best}$ {Remove the best candidate from the consideration set.}
if $\|CPC\| = \log_2(20 * ss)$ **then**
for all $c \in CPC$ **do**
if $\exists c \in CPC : c \perp v | CPC \vee \exists pc \in CPC : c \perp v | pc$ **then**
CPC $\bar{=}$ c {Remove any element in CPC that has become independent given the rest of CPC or now has a sibling relationship given another variable in CPC.}
end if
end for
if $\|CPC\| = \log_2(20 * ss)$ **then**
Move the last element of CPC into OPC {If our current Markov blanket is at the sample limit, move the most recently learned element to the overflow list.}
end if
end if
end while
$\forall c \in CPC \cup OPC$ set $v \sim c \in G$ {Store the discovered Markov blanket in the graph.}
$\forall c \in CPC_0 \backslash CPC$ set $v \not\sim c \in G$ {Remove any edges from the graph which were erroneously in the initial CPC.}
$L \leftarrow v$
end for
Return Reduce-TW(Greedy-Search(D,G),G)

We also used Tetrad's implementation of Grow-Shrink as a starting point for developing the Markov-blanket discovery portion of Program 7.5.

Tetrad internally stores its datasets as a dense in-memory matrix of 64-bit double values, which would take over two gigabytes of RAM for just the GopherInpatient dataset (without any intermediate calculations). In order to handle large datasets effectively, we wrote an

adapter for Tetrad to use sparse matrices of 32-bit floating-point values, from Wendykier and Nagy's (2010) Parallel Colt library. These were slightly slower than dense matrices, but because the entire Tetrad suite was modified to use them, our speed tests are consistent across algorithms.

In order to restrict searches to the MAG in MAG-GS and MAG-BS, we wrote a tool to convert a JUNG graph to a Tetrad graph, so the MAG (and subsets of it selected by FS-IMPORTANCE) can be loaded into Java using the methods developed in Chapter 6. Programs 7.5 and 3.1 both require a restriction graph, so we implemented BS and GREEDY by first generating the fully connected graph as the restriction graph. (The time taken to build the fully-connected graph was subtracted from the speed tests below.)

We performed two types of evaluation. First, we compared our flagship algorithm, MAG-BS, with existing algorithms. These included: GES, which we have used up to this point; unconstrained greedy search, GREEDY; MMHC, which is the best existing hybrid algorithm[3]; and CPC, which is a popular constraint-based algorithm. Second, we compared the three variations of our algorithm among themselves: MAG-GS,BS, and MAG-BS.

The primary dataset used for evaluation was a dataset of pregnancy-related hospitalizations chosen by the FS-IMPORTANT program (Chapter 6) from GopherInpatient (which we will refer to as GopherInpatPreg). Additionally, we compared our algorithm to the others using the Andes dataset used in Chapter 5 (to compare the algorithms in a less-sparse domain), and we performed a larger-variable evaluation of our algorithms on the full GopherInpatient dataset (the other algorithms did not scale well above 200 variables).

We wrote a Java program to select subsets of each dataset with increasing numbers of variables. As with the program in Chapter 5, in each iteration the program generates a data subset containing all samples but only the variables in the current iteration. The ordering of the variables for subset selection was as follows: GopherInpatPreg was ordered by importance, as chosen by FS-IMPORTANT (so the most highly correlated variables would be chosen in the smaller subsets), and GopherInpatient and Andes were randomly but consistently ordered.

[3]The version of MMHC in Tetrad does not contain all of the speed optimizations in Tsamardinos et al.'s (2006) paper.

We evaluated on three axes.

The first axis was speed, measured in seconds to learn the graph. To measure this, we timed the learning algorithm on every subset of the dataset. As in Chapter 5, the speed test begins after the data subset is created and all data structures are set up, so only the runtime of the learning algorithm are compared. Although the full MAG need only be learned once, for the speed evaluation we relearn the MAG on each subset of variables.

The second axis was accuracy in inference. For each subset, we used the average area under the receiver-operator curve (AUC) on a test set (1/3 of the data). To compute AUC, we converted our graph to SMILE format using our previously written converter and ran EVAL (Program 4.2) using the admission-compressed test set.

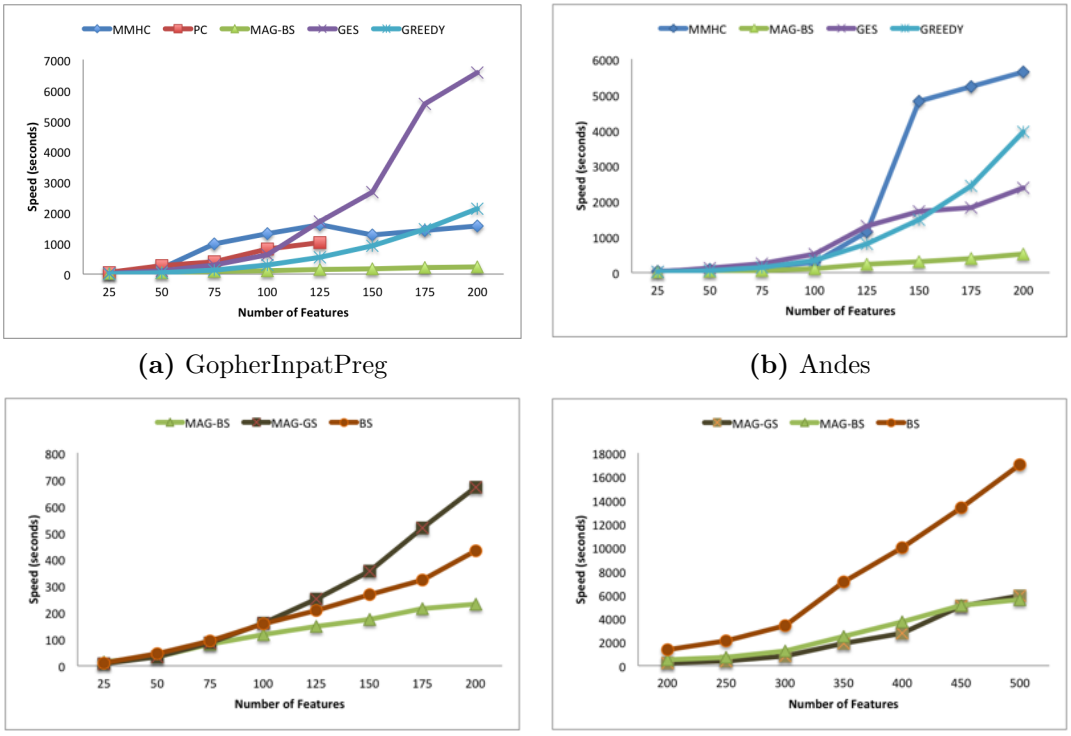
The third axis was graph complexity. A typical measure of graph complexity (e.g., in Tsamardinos et al. (2006)) is a comparison of the final graph structure to the generative graph. However, there is no generative graph for Gopher data and the original Andes graph could not be learned by any existing structure learning tool (because it involves connected subgraphs). Moreover, our complexity concern is not about similarity, but speed in inference. Therefore we compared the number of edges learned by each algorithm (which is loosely related to speed in inference).

To choose *max-tree-width* for GREEDY, we experimented with SMILE and found that it always performed very quickly in inference on graphs up to tree-width of 5. Moreover, we found the graphs learned on GopherInpatientPreg with other algorithms almost universally had tree-width of 3 or 4 (regardless of subset size). Finally, we verified during evaluation that bounding MAG-BS with a tree-width of 4 removed at most a few edges. Therefore we set *max-tree-width* to 4.

We performed the evaluation across algorithms on GopherInpatPreg and Andes on subsets of 25-200 variables with an increment of 25. We performed the evaluation among our algorithms on GopherInpatPreg and GopherInpatient on subsets of 200-500 variables with an increment of 50. We report the speed, AUC, and number of edges for each dataset subset with each algorithm.

3.2. Results. We found that all algorithms on all datasets performed similarly in AUC, so we report only the average ± standard deviation for each algorithm compared

to an unconstrained greedy search on each dataset, and only in cases where the average difference exceeded .01. This can be seen in Table 7.2. Network complexity did vary by algorithm. The complexity differences between MAG-BS and preexisting algorithms on the GopherInpatPreg and Andes datasets can be seen in Figure 7.3. The complexity differences among variations of our algorithm were smaller than any differences between algorithms, so they are not reported here. MAG-BS consistently produced the smallest networks of the three variations. Speed differences were large and are reported in 7.1. Note that PC frequently found graphs with cycles, and only when a DAG was found is it reported here.



(a) GopherInpatPreg **(b)** Andes

(c) GopherInpatPreg **(d)** GopherInpat

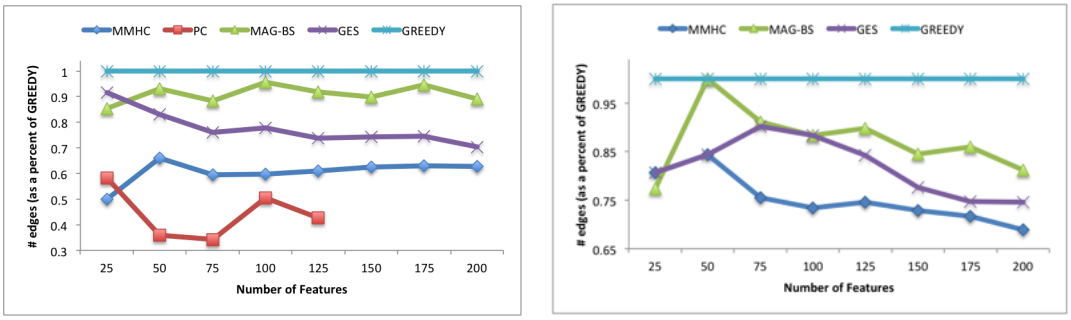
Table 7.1. Learning speed in seconds across algorithms and datasets. Top row: MAG-BS vs. pre-existing algorithms, on the GopherInpatPreg and Andes datasets, respectively. Bottom row: variations of our algorithm, on the GopherInpatPreg and GopherInpat datasets, respectively.

3.3. Discussion. MAG-BS was at least twice as fast as any existing algorithm for subsets of at least 100 variables on GopherInpatPreg and 75 variables on Andes. At 200 variables, MAG-BS was seven times faster than the next fastest algorithm (MMHC) on

GopherInpatPreg and four times faster on Andes. On both datasets, MAG-BS did find the second largest graphs after greedy search, but it tended to follow the next smallest graph by a constant percentage, indicating that the complexity penalty is linear in number of nodes.

	MMHC	PC	MAG-BS	MAG-GS	GES	BS
InpatPreg	$-.023 \pm .01$	$-.052 \pm .019$			$-.011 \pm .007$	
Inpatient	n/a	n/a			n/a	
Andes	$.01 \pm .026$	**X**		n/a	$.017 \pm .032$	n/a

Table 7.2. The average difference and standard deviation of average AUC from GREEDY among all algorithms on all datasets. Only differences that exceed .01 are reported. PC is not reported for the Andes dataset because it was unable to generate a graph without a cycle. Inpatient and InpatPreg correspond to the two Gopher datasets, GopherInpatient and GopherInpatientPreg, respectively.



(a) GopherInpatPreg **(b)** Andes

Table 7.3. Network complexity of discovered networks on the GopherInpatPreg and Andes datasets, measured as a percentage of the edge count found by an unconstrained greedy search.

Perhaps more importantly, the computational performance (in terms of AUC) was not affected. We were surprised to find that average AUC across all algorithms was so similar on both datasets. On the GopherInpatPreg dataset, an unconstrained greedy search slightly outperformed all other algorithms. This indicated to us that this domain is more sensitive to missing edges than finding a non-optimal I-map. In Andes, GREEDY performed worst, as expected. MAG-BS came in second, also as expected, because it is the only other algorithm not guaranteed to find the P-map. Again, however, differences in AUC were very small (largest difference $.017 \pm .032$).

Among our algorithms, MAG-BS ran by far the fastest on the GopherInpatPreg dataset on all subsets of at least 75 variables. On the GopherInpatient set, MAG-GS was very similar (and occasionally superior) in speed. This highlights the less correlated nature of the domain. Because the variable ordering was random on GopherInpatient, the time spent learning the skeleton in MAG-BS did not translate into a significantly reduced edge space during the GREEDY phase. All of our algorithms performed similarly in terms of computational performance and graph complexity on both datasets.

Because GopherInpatPreg is most like the datasets we will actually be using in the rest of this work (e.g., an order-entry dataset with feature selection performed), we believe that the conclusions drawn about that domain are most relevant to the rest of this work. In particular, at 200 variables MAG-BS is seven times faster than any existing algorithm, twice as fast as our other algorithms, and less than .01 different in average AUC. The tradeoff is slightly more complex graphs, but they are still more than 10% smaller than a greedy search.

3.4. Limitations and Future Directions. We cannot say definitively how much faster MAG-BS is than all existing algorithms, because we only evaluated against what was available in Tetrad. The implementation of MMHC in Tetrad is somewhat different than the original description. In particular, it does not have all the skeleton-generation computational optimizations, and it orients via a GES-based orienter rather than a greedy-search orienter. We have found that the GES-based orienter does not scale particularly well (though it does much better than pure GES). Therefore the comparison of MMHC's speed to MAG-BS is somewhat disingenuous. It could be better to compare MMHC's implementation in Causal Explorer to MAG-BS, although this introduces the confounding factor of an entirely different computing platform (MATLAB vs. Java). Also, the fastest constraint-based learner, Collider-Sets, is not available in any known implementation, so we used the best available to us, Conservative PC (CPC).

Another issue is that the speed and accuracy of a learner is, as we have seen, highly dependent on the characteristics on the dataset, and it would be wise to perform an evaluation on a wider variety of datasets. In particular, datasets of very complex relationships

in which greedy search performs much worse than algorithms which find a P-map should be tested, to measure more closely the impact of approximate skeletons on learning.

Third, allowing each blanket learned in BS to remove edges which already exist in the skeleton slightly decreases the stability of the algorithm. This is because in a dataset not faithful to some graph, the Markov blanket is not unique. Then the final skeleton discovered is dependent on the ordering in which the variables are added to the skeleton. This order is currently random and stability could be improved by adding them in some principled way (e.g., by choosing them in reverse order of degree in the MAG). This instability could be removed entirely by iteratively revising the Markov blankets of nodes impacted by this edge removal process. However, in the worst case this would greatly increase the time complexity of BS.

Finally, the choice of a maximum treewidth in our greedy search is currently chosen through experimentation rather than through a reliable statistical process.

CHAPTER 8

Probabilistic Temporal Models

In this section, we have been developing methodologies for a robust and rapid Bayesian-network-based experimentation platform to extract local wisdom from decision support data. In the previous chapters, we designed and evaluated a principled and efficient method to select and learn domain-specific Bayesian networks. One troubling problem remains: networks encode variable associations that are time-agnostic. Probability flows in both directions in the graph.

The bidirectional flow of probability is essential to inference, as discussed with regard to Figure 5.1. However, it does not always result in appropriate inferential conclusions. For example, in the abdominal pain network in Figure 2.3, not only can a pregnancy test lead to an obstetrics consult, but an obstetrics consult can lead to a pregnancy test. This is illustrated in Figure 8.1. This is likely not our meaning in this situation.

The trouble has to do with time. We do not mean for an obstetrics consult to trigger a pregnancy test, because one always must occur before the other. Therefore we need a method for dealing with temporality in our networks. Fortunately, several such approaches exist. Unfortunately, none of them are particularly mature, most greatly increase the computational burden on both inference and learning, and almost none are available in existing software toolkits.

Therefore in this chapter we first review existing methods and existing tools for implementing time in Bayesian networks. Then, we design an innovative method which is both computationally simpler than existing methods and very powerful for our task. Finally, we implement and evaluate this method on the domain of inpatient pregnancy.

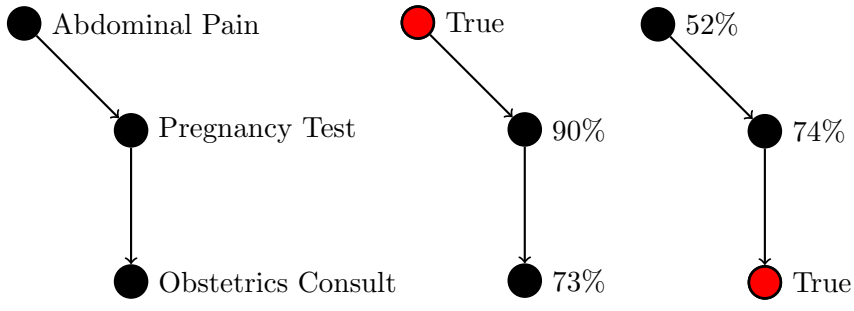


Figure 8.1. An example of bidirectional probability flow in a BN.

1. Review of Temporal Methods

Reasoning about time is critical in many domains. In a monitoring network, it may be important to predict how long after sensors enter a configuration a given event occurs. In a diagnostic network, one might want to reason about what might have caused a car to stop running given the current lights on the dashboard. The first task (looking forward in time) is known as *prediction*, whereas the second (looking backward in time) is known as *smoothing*. Murphy (2002) gives a good overview of these temporal queries.

In this section, we limit our review to one very specific prediction task: *what happens next?* That is, given a patient in a set of given past states ($P_{\{t...(t+n)\}}$), we would like to know what actions are performed within the CPOE at time $t+n+1$ (P_{t+n+1}). This is not to say this is the only relevant prediction task. For example, we might in some circumstances have interest in what happens later in the hospitalization, for example, how likely is pneumonia to lead to intubation? It might also be interesting to ask longer-term questions, such as how likely a patient with CHF is to be rehospitalized within a month. Looking backwards in time could also be beneficial for data quality queries, e.g., if we ordered a pregnancy test, what is the likelihood we should have added abdominal pain to the problem list? However, each of these tasks have different goals and require distinct temporal-reasoning capabilities. Therefore here we review the available options with an eye to the goal of this research.

Because the algorithms can be very complex, it is important to consider available tools. Therefore we also review available software tools to perform inference and learning in these motifs.

1.1. Causal Bayesian Networks. In general, the edges and directionality in a network have a statistical meaning that does not strongly correlate with common sense. For example, in score-based learning, edges only contribute to the likelihood that the graph represents the underlying statistical distribution of the data. However, if we could somehow learn a network where the edges had a commonsense meaning, where a node's children were its direct effects, then we would have one of Pearl's (2000) Causal Bayesian Networks (CBNs).

It now becomes possible to reason about causes. In particular, in the abdominal pain network above, we recognize that ordering an obstetrics consult is an *intervention* and not

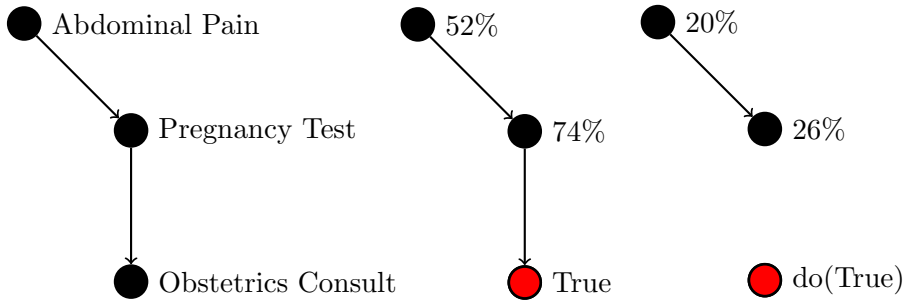


Figure 8.2. An example of inference in a Causal Bayesian network.

a mere observation. Pearl introduced a do() operator for this situation. If a physician orders an obstetric consult, this is do(Obstetric Consult=1). We are now no longer inclined to change our beliefs about why the obstetric consult happened. Perhaps the physician performed a pregnancy test or perhaps she wanted to increase the revenue of the obstetrics team. Either way, we temporarily delete the incoming edges of a node that has had a do() performed, because we no longer are concerned about its cause. This is called *mangling* the graph, and is shown in Figure 8.2.

In fact it is possible to learn such a graph from data. The constraint-based learning algorithms discussed in Chapter 6 frequently do a good job of orienting the graph with regard to its actual causes (Spirtes et al., 2000), so when causality is more important than stability these can be a good choice. Better, if interventional data are available, Cooper and Yoo (1999) developed a methodology to learn CBNs directly with any learning algorithm. The solution is to introduce a asymmetry in the parameter counts, so that $M(A,B) \neq M(B,A)$ when interventions are involved.

The free Pebl toolkit supports this interventional learning, and it offers a scalable Python-based learning engine which can perform structure learning through a best-first greedy search or simulated annealing learner. Implementing the do() operator is fairly straightforward and can be done on any package that offers an application programming interface (API) to manipulate the network. We review such packages in Appendix A.

Because CPOE data is interventional, CBNs might seem like a good option for time representation. The patient's state at time $t+1$ is the result of interventions made at time t. To learn such a network, we might create a dataset with one row per order session, where

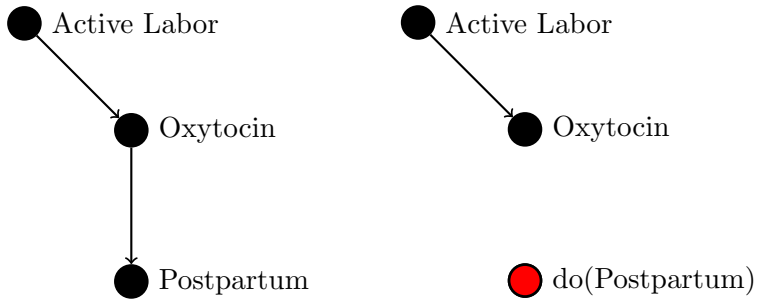


Figure 8.3. An example pregnancy network before and after do(Postpartum). In this case, we would like Oxytocin to remain related to Postpartum after the intervention, as it is also a postpartum treatment.

each event from a previous time is an intervention and each event in the current time is an observation.

The problem with this approach is that *only* interventional relationships are supported in CBNs. Two nodes cannot have both an interventional and non-interventional relationship. However, nodes frequently have both relationships. In the pregnancy domain, oxytocin both induces labor and reduces postpartum hemorrhaging. After do(Postpartum), the relationship with oxytocin will be severed, but we would like it to still exist as a postpartum treatment option. This issue is is illustrated in Figure 8.3. We would like to instead have both a causal and non-causal edge between postpartum and oxytocin. This type of structure is supported in Dynamic Bayesian Networks (DBNs).

1.2. Dynamic Bayesian Networks. The Dynamic Bayesian Network (DBN), introduced by introduced by Dean and Kanazawa (1989), defines two edge types, temporal and non-temporal. Such networks are often represented by two copies of the network for a given time, connected by temporal edges. This is called a 2-TBN (two-slice temporal Bayesian network). The temporal edges exist from t to $t+1$ and define an inter-slice topology. Then, non-temporal edges within each slice define regular BN relationships, an intra-slice topology. Both copies of the intra-slice topology are the same. DBNs are reasoned over by 'unrolling' the time slices into a large network containing many copies of the basic network. The cost is doubling the number of nodes in the network.

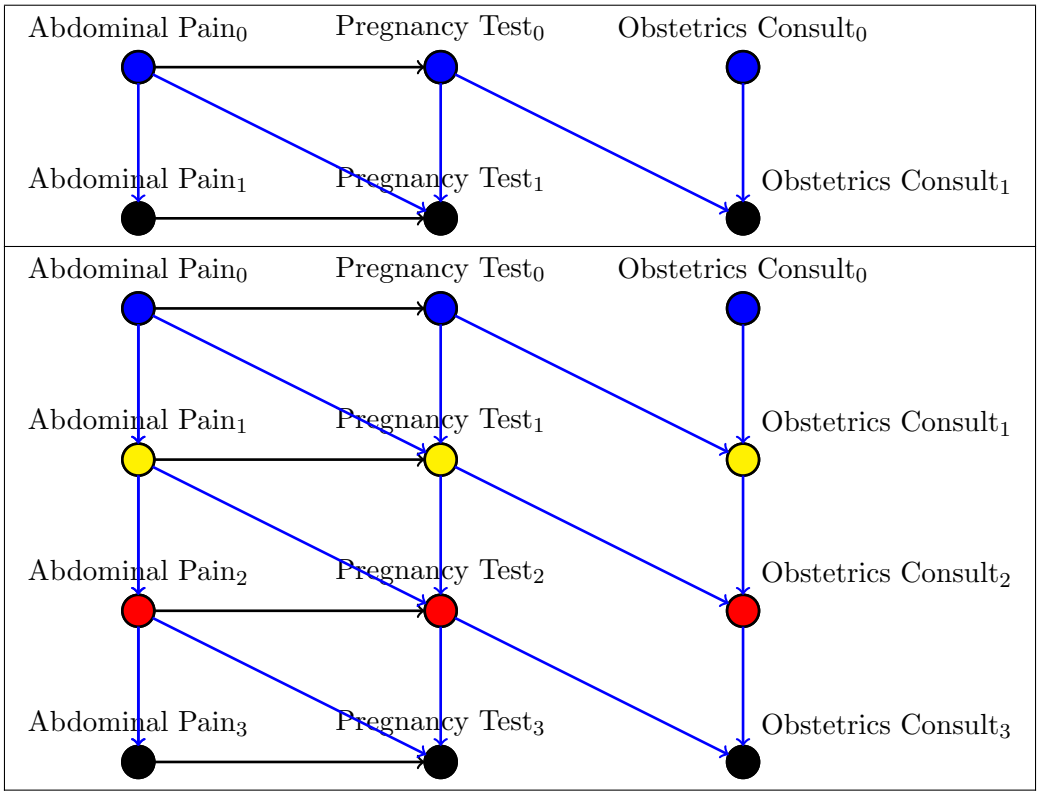


Figure 8.4. An example of Figure 2.3 as one possible Dynamic Bayesian network (top), unrolled into three time slices (bottom).

Figure 8.4 shows Figure 2.3 as one possible DBN, both in 2-TBN and unrolled format. It separates the temporal from non-temporal edges in the network, placing the transition from pregnancy test to obstetrics consult in the temporal tier. Here the notion of present and past are separated, so e.g., an obstetrics consult will not trigger a pregnancy test (though it would indicate that one ought to have occurred). By unrolling the network, we can reason about future events based on an arbitrary number of steps in the past. In this example, there are also temporal edges from each variable to itself. This is common in DBNs and means that the likelihood of a node at one time has impact on the probability of the node at the next time. Therefore a diagnosis of abdominal pain at t_0 impacts the probability of abdominal pain at time t_3. This allows temporal decay of evidence at any rate that fits an exponential curve. It does not allow explicit relationships of variables to time (e.g., no obstetrics consults happen after midnight), but this is frequently unnecessary if the appropriate choice of proxy variables are chosen.

DBNs offer a compact way of modeling any system that has a consistent, indirect relationship with time and in which the future is independent of the past given the present (e.g., once inference has completed, it is permissible to 'roll up' the 2-TBN again.) Unfortunately, DBNs also have significant limitations.

One limitation is complexity in inference. Notice that in Figure 8.4, every variable is correlated with every other variable across time slices. This is called *entanglement* and frequently occurs even when d-separation relationships are present in the rolled up network. It renders DBNs intractable for inference after only a few time slices. At present, the most common solution in existing software is to limit the unrolling depth of the DBN. Other (complex) options exist in research. Boyen and Koller (1998) introduce an algorithm (BK) which causes old inference information to 'vanish' exponentially, rolling up the network periodically into the most likely evidence. Therefore the inference problem is never more than a few time-slices. This appears to be a good solution, although only Murphy et al.'s (2001) Bayes Net Toolbox (BNT) implements it (that we are aware of).

A second limitation is complexity in learning, which is an ongoing research problem (Friedman et al., 1998). Because all parameters are frequently not fully observed in each time slice, expectation maximization is used to guess the missing values, which is far slower than the maximum-likelihood approach (see Chapter 2). Structure learning of DBNs is also a difficult problem. The only tool we are aware of that offers an approach is, again, BNT. For the case of a fully-observed time series of binary variables, BNT implements Liang et al.'s (1998) REVEAL algorithm to learn the inter-slice topology. This considers all possible combinations of temporal parents, rather than a greedy search heuristic, and therefore requires $O(n^{k+1})$ time, where k is the maximum fan-in.

A third limitation is the discrete nature of time. It is often not reasonable to observe a system at consistent time intervals, and even if such an undertaking is possible, many variables evolve at different rates. For example, blood pressure might be observed hourly, but a patient's level of pain might only be assessed daily. It is not possible to support two granularities of time in a single DBN.

1.3. Continuous Time Bayesian Networks. Primarily in response to the third limitation above, Nodelman et al. (2002) introduced the Continuous Time Bayesian Network

(CTBN). This model reduces the transitions from $t \to (t+1)$ to $t \to (t+\epsilon)$, so every edge is an instantaneous temporal edge. Again, CTBNs do not allow variables to be explicitly dependent on time (each variable is a function of other variables, not directly time). Like DBNs, the network evolves over time. In a CTBN, each variable is modeled as a Markov process that depends on its parents.

Although it is appealing that the CTBN does not have both inter- and intra- slice edges (because all edges are instantaneous temporal edges), inference and learning is more complex than even a DBN. Rather than unrolling the network for k discrete time slices, the reasoning engine simulates the evolution of each variable from time 0 to k. Exact inference is intractable, but Nodelman et al. (2005) have developed an approximate inference algorithm which runs in bounded exponential time. It has the appealing characteristic that it automatically chooses the correct time granularity to reason about. Structure learning of CTBNs, according to Nodelman et al. (2003), could be somewhat faster than DBN learning, because the graph can contain cycles (and so time-consuming cycle checks can be eliminated). The authors have developed a Bayesian score for CTBNs that can be used by score-based searches.

CTBNs are an appealing methodology, and the principal limitation currently is lack of support and integration with other Bayesian network methods. They are more reminiscent of Markov processes than Bayesian networks and therefore cannot directly take advantage of the many advances in Bayesian learning and inference (e.g., hybrid algorithms in structure learning). Also, only one tool, Shelton et al.'s (2010) CTBN-RLE, supports CTBNs currently.

1.4. Modeling Time Explicitly. Both DBNs and CTBNs make the assumption that variables are functions not of time but of the previous state of those variables. Therefore they cannot model situations where variables depend directly on time.

Tawfik and Neufeld (1994) model time explicitly. The probability of each node state is a convolution of the time functions of each of its parents. This allows reasoning over a network of discrete variables, but now probabilities propagate according to associated temporal functions. Because the number of time functions is exponential in the number of parents, Tawfik suggests defining a single function ϕ on each variable that captures the

impact of all its parents as time evolves. Although Tawfik describes important properties ϕ must hold, he does not explore methods to learn the function (nor how to learn the structure of the network). He proposes this as future work, but to our knowledge it has not been pursued.

1.5. Summary. We have reviewed methodologies for handling time in Bayesian networks, for which there exist a dearth of available tools. CBNs involve relatively modest extensions to existing methods and can learn and reason about networks in which every edge is causal - i.e. the parents of one node are its direct causes. DBNs are a more flexible temporal approach that can model discrete time and exponential decay, but they are subject to complexities not found in typical Bayesian networks - especially entanglement (the super-correlation of all variables in the network) and the open problem of efficient structure learning (especially when every time slice is not observed). CTBNs are a newer approach that model continuous time as a network of Markov processes. These are not quite Bayesian networks but share some of their properties. Inference is quite complex and can be slow (because it requires simulation of a continuous process), but structure learning might be efficient due to the cyclic nature of the networks. Alternatives which model time explicitly (rather than as functions of variables in earlier times) have only been tersely researched.

All of these models are complicated to implement, so it is important to consider available software (which can be used as building blocks) when developing a temporal approach. The capability for learning and inference on these three models using available software is shown in Table 8.1. Note that inference in CBNs is straightforward to implement in packages with APIs, though their structure is not (at least not without source code). Also, several packages support limited inference in DBNs by unrolling a fixed number of time-slices. BNT alone has full support for DBNs, and it is limited by an implementation which is not scalable, relying on inefficient Matlab cell arrays.

2. A Tractable Temporal Model for our Application

In this section, we develop a scalable temporal model that is geared toward the prediction of actions and can be implemented relatively easily using existing tools. In our CPOE data, observations (orders) occur in sessions. This discrete-time system is most closely

	Netica	BayesiaLab	BNT	Pebl	SMILE	CTBN-RLE
CBN	-	I,S	I,S	S	-	-
DBN	I*	I*	I,S	-	I*	-
CTBN	-	-	-	-	-	I,S

Table 8.1. Software tools for learning and inference on temporal models: a summary of features available in popular packages. I = inference, S = structure learning. * = limited capability

approximated by DBNs, so our model builds from that base. First we develop a type of DBN more appropriate to predicting actions (the temporal-causal DBN), and then we simplify the model to allow inference without unrolling (the time-forward temporal-causal DBN). We develop both approaches using binary variables to simplify the discussion, but it is straightforward to support multinomials.

2.1. Temporal-Causal DBNs. DBNs separate temporal edges from intra-time-slice edges, and this allows DBNs to support both temporal and associative relationships between variables. However, the temporal edges still conflate two meanings. Temporal edges can be *causal* (e.g., an event at time $t-1$ precipitates an event at time t) or *fluent propagation*. Hayes and McCarthy (1969) call a *fluent* the tendency of variables to remain in the same state over time, until a new observation contradicts this belief. These fluent edges are the self-edges in Figure 8.4. They not only clutter and complicate the DBN, but they result in incorrect conclusions. A network with both fluent and causal edges will predict actions at time t either if they are caused by the state at $t-1$ or if the state remains unchanged at time t. For example, in Figure 8.4, the probability of a pregnancy test is positively associated with both the probability of abdominal pain (causal edges), and on the probability of a previous pregnancy test (a fluent edge). We desire a network that predicts actions at time t, not the truth state of variables.

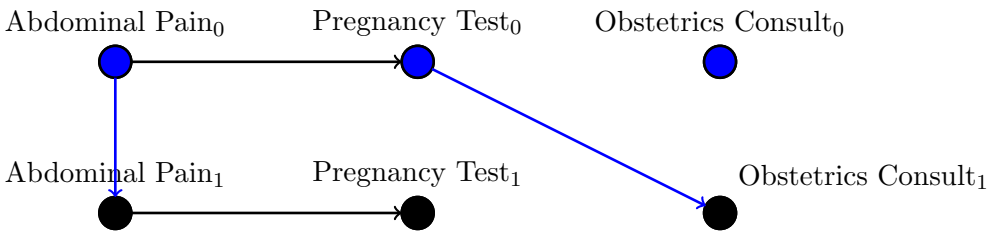
A structure learning example is illustrative of this problem. We designed Figure 8.4 by hand, but a statistically learned network might favor the fluent edges over the causal ones. Although of course such a network will still predict what the data indicate, it illustrates how the data are not what we intend. An example of this problem can be seen in Figure 8.5. First, a source dataset is shown. It contains two time-series of variables with three states: observed true (1), observed false (0), and unobserved (?). In the first time-series, abdominal

Abdominal Pain	Pregnancy Test	OB Consult
1	?	?
?	?	?
?	1	?
?	?	1
1	1	?
?	?	1

(a) Source dataset.

Abdominal Pain$_t$	Preg. Test$_t$	OB Consult$_t$	Abdominal Pain$_{t-1}$	Preg. Test$_{t-1}$	OB Consult$_{t-1}$
1	0	0	0	0	0
1	0	0	1	0	0
1	1	0	1	0	0
1	1	1	1	1	0
1	1	0	0	0	0
0	0	1	1	1	0

(b) Temporal dataset.



(c) Resulting graph.

Figure 8.5. A typical DBN conflates fluents and causes in the temporal tier, resulting in structures that predict truth-states rather than actions. Top: two time-series of a source data set. Middle: a conversion of the data into a representation suitable for a traditional DBN. Bottom: the resulting network, as learned by GES or BS.

pain leads directly to a pregnancy test and therefore to an OB consult in the next session. In the other, one event occurs per session and a session exists with no events. Next, a version of this dataset with two time-slices (t and $t-1$) is shown. By the fluent property, we assume that all observations remain true once they become true. (Observations could expire after some time, but the idea is the same.) Therefore a variable in t is true in the target dataset if it was last observed true in the source dataset, and a variable in $t-1$ is true if its equivalent in t was true in the previous row. The resulting graph is shown (learned using BS). Notice that Abdominal Pain$_{t-1}$ influences only Abdominal Pain$_t$ and our causal edge to Pregnancy Test is lost. The 'meaning' of this network (and therefore the underlying data) is that Abdominal Pain in the past has a stronger influence on Abdominal Pain in

the future than it does a Pregnancy Test. This does not make sense in an action-oriented predictive network.

Therefore we introduce a class of networks that allow only causal edges in the temporal tier: the temporal-causal dynamic Bayesian network (TCDBN). Fluent edges are replaced by a deterministic function embedded in each $t-1$ node which determines how likely it remains true into the next time slice.

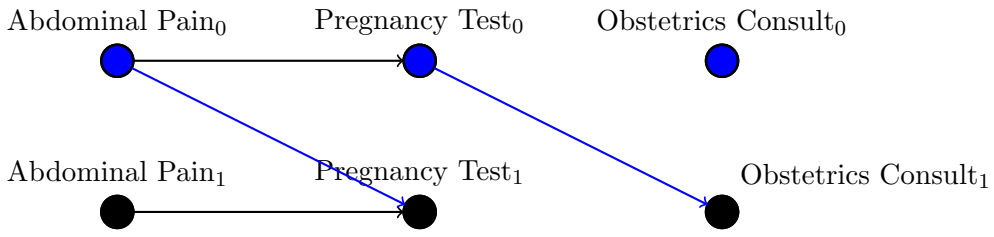
Definition 7. *A temporal-causal dynamic Bayesian network (TCDBN) is a DBN in which only causal edges are allowed in the inter-slice topology, and fluency is handled through a deterministic function embedded in each $t-1$ node, called a decay function.*

Abdominal Pain	Pregnancy Test	OB Consult
1	?	?
?	?	?
?	1	?
?	?	1
1	1	?
?	?	1

(a) Source dataset.

Abdominal Pain$_t$	Preg. Test$_t$	OB Consult$_t$	Abdominal Pain$_{t-1}$	Preg. Test$_{t-1}$	OB Consult$_{t-1}$
1	0	0	0	0	0
0	0	0	.8	0	0
0	1	0	.64	0	0
0	0	1	.512	.8	0
1	1	0	0	0	0
0	0	1	.8	.8	0

(b) Temporal dataset.



(c) Network.

Figure 8.6. A TCDBN, which does not conflate fluents and actions. This is learned from the same source dataset in Figure 8.5, converted to a temporal dataset using Program 8.1, and then fed to a version of MAG-BS modified for fractional counts in order to learn graph structure.

The first step in converting our source dataset in Figure 8.5 to TCDBN data is to put the fluents in the $t-1$ slice but not in the t slice. Then the t slice represents only the probability that a node becomes true during that time slice.

Next, the truth values in $t-1$ are replaced by a deterministic function. This solution is reminiscent of Tawfik's (1997) work, which embedded functions of time in the CPT of each node (discussed previously). By replacing observations with a deterministic function, we are able to model an important piece of metadata. Not only are past observations less likely to remain true as more time passes, they are also likely to decay in *relevance*. This allows us to model explicitly the fact that the strongest predictors of an order should be the most recent ones. Jensen and Nielsen (2007) apply this concept to inference by dynamically adjusting the conditional probability tables. They apply an exponential decay function to the counts in CPTs as time passes to adjust the influence of a variable during inference. They call this *fading*, which they use to adjust the influence of a variable during inference. We borrow this concept and apply it to every observation during the learning process.

Our method is illustrated in Figure 8.6. With the source dataset from Figure 8.5, we create a target dataset with two time-slices. We set all columns in t to 0 unless it was observed to be true in the source dataset. In each variable in $t-1$, we set the value to a decayed value as follows. If the variable was observed in this row, unobserved in this time-series, or most recently observed false, the value is zero. Otherwise, the decayed value is ρ^{t_Δ}, where ρ is a decay rate and t_Δ is the difference between the current time and the most recent observation in which it was true in this time-series. This method is outlined in Program 8.1. In Figure 8.6, $\rho = .8$.

Choosing ρ is an example of the *frame problem*: how long does something, once observed, remain true? (Hayes and McCarthy, 1969) The formulation here limits ρ to a single value. It is of course possible to specify a ρ for each node. However, choosing feature-specific ρ empirically is unfortunately difficult on Gopher data. Although it is possible to discontinue orders and diagnoses, this is infrequently done in practice, so discontinuation by itself is not a good cue for determining decay rate. We have done work on learning survival functions for diagnoses using multiple cues in the Gopher data (Klann and Schadow, 2010), and though we found distinguishable differences between long- and short-term diagnoses, we could not

extract a meaningful length-of-diagnosis. For this work, then, we choose a single ρ through experimentation.

Program 8.1 Temporalize-dataset(D_s,I_s,ρ)

Input: D_s is a source dataset of columns of observations. Valid values are 0 (observed false), 1 (observed true), and ? (unobserved).
Input: I_s is a list of start, stop indices for each time-series in D_S.
Input: ρ is a decay rate between 0-1
D_t is a temporal dataset initalized to all zeros, with the same number of rows as D_s and twice the columns.
n_c is the number of columns in D_s
for all time-series in I_s **do**
  for all rows r in this time-series (with indices starting at $i = 0$) **do**
    for all columns c in D_t **do**
      if $c \leq n_c$ **then**
        $D_t[r, c] = (D_s[r, c] == 1)?1 : 0$ {Copy D_s into the first half of D_t column-wise}
      else
        t_Δ = tidx$(D_s, c - n_c, I_s) - i$ where tidx(D, c, I_s) is a function that returns the index of the last time column c in dataset D was true within this time series I_s, or 0 if it was more recently observed to be false than true.
        if $t_\Delta > 0$ **then**
          $D_t[r, c] = \rho^{t_\Delta}$ {Compute time decays for for the second half of D_t}
        end if
      end if
    end for
  end for
end for
Return D_t

This temporal dataset now has fractional values for observations, so we need a learning algorithm that can learn discrete CPTs with fractional (i.e. uncertain) observations. We will show in a later section that is methodologically fairly straightforward and only requires modification of the BDeu statistic and G^2 test. With such fractional modifications, all that is left to do is force the orientation of edges from time $t - 1$ to point toward time t. Then any traditional structure learning algorithm can be used, including the suite of algorithms developed in the last chapter. The graph at the bottom of Figure 8.6 was learned from the dataset at the top of the figure using the BS algorithm after adjusting it for fractional counts and edge restrictions. It captures the expected temporal-causal relationships.

2.2. Time-forward TCDBNs. TCDBNs allow us to more accurately predict actions by separately specifying the truth state of variables. However, they still require unrolling in

inference. Therefore in this section we make two simplifying assumptions that greatly reduce complexity: one, we are only interested in predicting future treatments; two, we assume that all relevant past variables were observed at some point. Therefore it is sufficient for temporal arrows to only propagate evidence forward. We call this a *time-forward TCDBN* (or TF-TCDBN).

Definition 8. *A time-forward TCDBN, or TF-TCDBN, is a TCDBN in which temporal arrows only propagate forward in time.*

TF-TCDBNs have the interesting property that this temporal asymmetry prevents entanglement, because although t is D-connected to all $t-n$, all $t-n$ are D-separated from every $t-n+k$.

Theorem 5. *TF-TCDBNs are not subject to entanglement.*

Moreover, the probability of a node at time t having a particular value is dependent on only the probability distribution of the nodes at $t-1$. All previous time slices can be discarded. Therefore if we could maintain a probabilistic state of variables at time $t-1$, then inference could be rolled up (as in the BK algorithm) after every time slice, and only two time slices would ever be necessary. Furthermore, because we assume that all relevant past variables have been observed, the probability distribution of time $t-1$ is entirely determined by the network, observations made prior up to time t, and the length of time since those observations. This is because in a TCDBN the state of each observed node in the $t-1$ slice is a deterministic decay function applied to the last observation.

Theorem 6. *In a TF-TCDBN, probabilities at time t are only dependent on the most recent observation for all nodes and the time since each observation.*

All this leads to a simple inference algorithm for TF-TCDBNs, in which we iteratively infer and then observe. We represent our TF-TCDBN without an intra-slice topology in the $t-1$ tier, and we perform inference by propagating observations made at every step into the past. This works as follows. We initialize the $t-1$ layer of the network to all zeros. Then we perform inference, record observations (i.e. evidence is set, either by a user or

Program 8.2 Infer-TFTCDBN(G)

```
Input: G is a TF-TCDBN
  S_t is a probability distribution of the states of variables at time t.
  S_{t-1} is a probability distribution of the states of variables at time t-1, initialized to all
  zeros.
  for t = 0 → n do
    S_t=Inference(G|S_{t-1})
    Make observations
    for all variables v with observations made at time t do
      S_{t-1}[v]=S_t[v]
      Clear the evidence v
    end for
    for all nodes n in time t-1 not observed at time t do
      if an observation exists for n then
        the probability of that observation is reduced by a factor ρ
      end if
    end for
  end for
```

some heuristic). Finally, we propagate the probability distribution found in the t layer to $t-1$. This propagation follows our general idea for converting data in Program 8.1. If a node in the t layer has evidence set, we move that evidence to the equivalent node in the $t-1$ layer. For all other nodes observed positive most recently, we reduce the probability in the $t-1$ layer by a factor of ρ. The inference algorithm is shown in Program 8.2. It requires an inference algorithm that supports uncertain observations. This can be achieved through *virtual evidence*, which introduces observations in hidden nodes to make the given probability distribution work out properly (Pearl, 1988, sect 2.2.2). We gain some insight into how probability propagates in Figure 8.7. This shows an example of inference in the graph in Figure 8.6 viewed as a TF-TCDBN, with parameters learned from the temporal dataset shown there.

Figure 8.7 shows that inference suggests correct actions in t, properly suggesting a pregnancy test when a current or past abdominal pain diagnosis is present, and suggesting OB consult only if a pregnancy test has occurred in the past. Fluents are not propagated via the network. One remaining issue is that actions continue to have high probability even if they occurred in the past and no new evidence calls for them. We will present a simple method to bypass this problem in the implementation below.

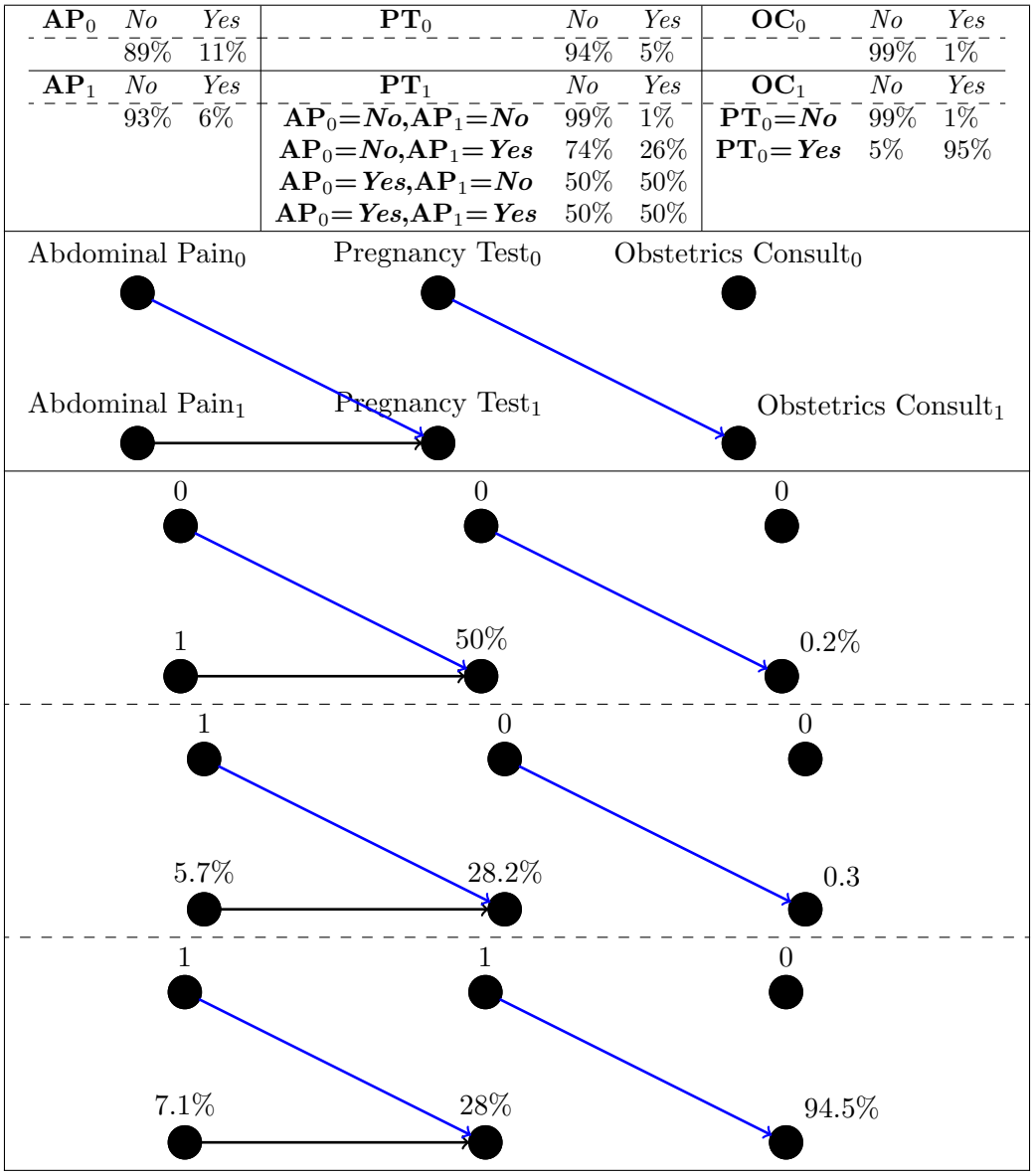


Figure 8.7. An example of inference in a TF-TCDBN. CPTs are specified for the graph in Figure 8.6 shown as a TF-TCDBN. An iterative inference example is shown with $\rho = 1$.

3. Implementing and Evaluating the Time-Forward Temporal-Causal Dynamic Bayesian Network

3.1. Implementation. In order to modify the BDeu and G^2 scorers to use fractional counts on binary variables, we required a Bayesian learning package with source code. We reviewed the features of structure learning packages with source code and show them in Table 8.2. Pebl did not support constraint-based learning. Also, it implemented a version

	BNT	Tetrad	Pebl	bnlearn
Constraint learning	Y	Y	N	Y
Score learning	Y	Y	Y	Y
Restricted edges	Y	Y	Y	Y
Source code	Matlab	Java	Python	R

Table 8.2. A summary of the features of open-source structure learning packages.

of BDeu known as K2 that could not be modified for fractional counts because it relies on factorials. Therefore Pebl was not considered. BNT was not considered for the scalability reasons discussed earlier. bnlearn and Tetrad were fairly equivalent in terms of features we needed, though neither directly supported DBNs. We chose Tetrad due to our familiarity with the Java language.

We modified the implementation of Tetrad by altering the code to compute counts of cells in a table to sum fractional counts, by treating fractional values in the datset as partial counts toward an observation of true. This was changed in the G^2 test and the BDeu scorer. We also modified MAG-BS to disallow directed edges from time $t-1$ to t and within time $t-1$, using Tetrad's edge restriction API.

Even though the network now supports multiple occurrences of the same action, the probability does not generally take past actions into account, so we adopt the same tactic as TreatmentSuggest. If an action has been performed, it does not reappear on the suggestion list. Therefore only actions with an average repetition of < 2 are considered in our evaluation.

We integrated the iterative inference loop from Infer-TFTCDBN (Program 8.2) into TreatmentSuggestInteractive. The modified program, TreatmentSuggestInteractiveTemporal, is shown in Program 8.3.

We implemented Program 8.1 as a SQL script and Program 8.3 using SMILE in Java. Because the Java API for SMILE does not support virtual evidence, only $\rho = 1$ is supported for inference.

The new treatment suggestion algorithm, which operates on order sessions rather than whole hospitalizations, necessitated a slightly modified GUI, which contains a 'Next Session'

Program 8.3 TreatmentSuggestInteractiveTemporal(G)

Input: G is a TF-TCDBN Bayesian Network Model
E is a set of evidence, where initially E includes all nodes in the $t-1$ layer set to 0.
O is a set of possible orders, which initially includes all orders in the t layer.
repeat
repeat
Update beliefs {Compute the posterior probability of all O $\notin$ E.}
Create a list of all O $\notin$ E in descending order of posterior probability, stopping at an optional threshold.
Remove orders from O if its $t-1$ pair $\in$ E.
Display the list to the user and wait for the user to choose an order from the list.
Move the order from O to E.
until the user finishes the session.
Move all evidence in E from the t layer to the $t-1$ layer.
for all nodes n in time $t-1$ not observed at time t **do**
if an observation exists for n **then**
the probability of that observation is reduced by a factor ρ
if $\rho = 0$, remove n from E
end if
end for
until the user quits the program.

button to propagate the probabilities to the $t-1$ layer. The new GUI running Program 8.3 can be seen in Figure 8.8.

Finally, we modified Program 4.2 (Eval) to also incorporate these changes. At the start of each hospitalization, we initialize the $t-1$ layer and all problems to 0. Then we perform evidence propagation at the end of each session. We again do not consider actions previously performed.

3.2. Evaluation Methods. We selected two domains of medicine from Chapter 4: the one that performed best (inpatient pregnancy) and the one which performed worst (the medical ICU). We performed a comparison of the performance of networks in these domains on temporal and non-temporal networks. This was achieved with following steps for each domain:

- We chose features by applying FS-IMPORTANT to the GopherInpatient dataset to select the top 40 orders and top 10 co-occurring diagnoses in each domain. We extracted all hospitalizations involving these 50 features.
- We split these data into a training set (2/3 of hospitalizations) and test set (1/3).

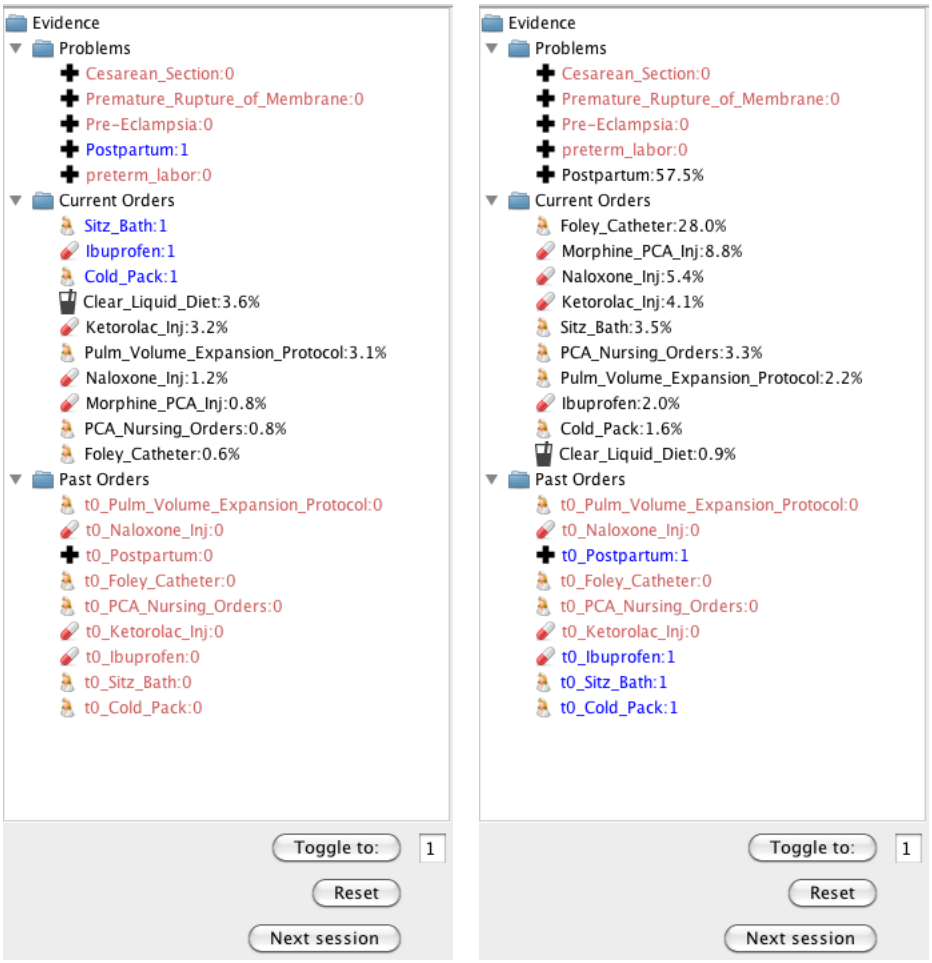


Figure 8.8. A GUI for TreatmentSuggestInteractiveTemporal. This is like Figure 4.1, except it includes the state of the previous session and a 'next session' button to propagate probabilities to the next session. Shown: before (left) and after (right) 'next session' is clicked.

- From the training data, we generated a baseline set D_B and an experimental set D_E. D_B was an admission-compressed table (as in Chapter 3). D_E was generated by Program 8.1 (Temporalize-dataset) with $\rho = .8$.
- To generate BNs for these datasets, we used our modified MAG-BS algorithm described above. Perhaps because the t-1 layer is never uncertain, we found larger *max-treewidth*s to be efficient on D_E. We set *max-treewidth*=5 on both baseline and experimental sets (so as not to unfairly favor the temporal graphs).

- We used our (previously written) converter to produce a SMILE network, and we applied our updated version of EVAL to compute PPV80 and tAUC on D_B and D_E.

3.3. Results. We report the differences in PPV80 and tAUC between the temporal and baseline networks, for the domains of inpatient pregnancy and the medical intensive care unit (MICU). Figure 8.3 shows summary statistics (unweighted average tAUC and PPV80). We use an unweighted average (unlike in Chapter 4) because, although it is a less accurate measure of the average use-case of the system, it is more affected by wide variance in performance. Histograms showing the differences in PPV80 in each domain are shown in Figure 8.9. The orders with the best and worst change in tAUC and PPV80 for each domain are shown in 8.4.

Domain	D_B avg(tAUC)	D_B avg(PPV80)	D_E avg(tAUC)	D_E avg(PPV80)
Pregnancy	.907	5.88	.921	4.02
MICU	.700	17.35	.733	15.47

Table 8.3. The (unweighted) average tAUC and PPV80 on baseline (D_B) and temporal (D_E) networks in two domains.

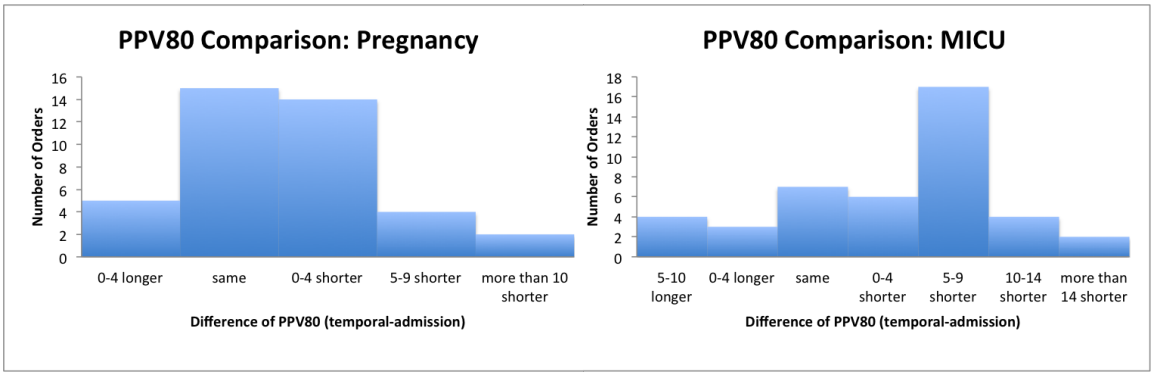


Figure 8.9. Histograms of the difference in treatment suggestion menu length given a positive predictive value of 80% (PPV80), for each order for each network.

3.4. Discussion. On average, the temporal networks performed only moderately better than the non-temporal baseline. On average, there was an increase of .02 in average tAUC, and a shortening of list length by 2 (by PPV80). However an inspection of individual

orders tells a different story. Many orders appeared in shorter PPV80 lists (20 in pregnancy and 26 in the MICU), while few appeared later (5 in the pregnancy, 7 in the MICU). The average shortening of the lists was greater than the average lengthening (12.5 vs 3.05).

The temporal network made more of an improvement in the MICU domain than in pregnancy. This is probably because the MICU has more frequent explicitly temporal relationships. For example, the order with the largest decrease in PPV80 in the MICU, Hang Blood, has a haptoglobin serum test as a parent in the $t-1$ layer in the learned network. This order, like our example of pregnancy tests leading to obstetrics consults, is not bidirectional. In pregnancy, bidirectional relationships are more typical (such as the variety of postpartum adjuncts).

Figure 8.9 reveals two general trends. PPV80 decreases are more common in the less frequent orders, whereas tAUC increases are more common in the most frequent orders. The former is likely because temporal relationship is important in uncommon orders. The latter could be true because temporal relationship distinguishes *when* the common orders are placed, but PPV80 still does not correlate well with tAUC in these cases for the reasons discussed in Chapter 4.

Overall, although the temporal networks increase average performance moderately, they successfully unmask temporal relationships in the data and improve performance where relationships are not bidirectional. The greatest benefit is seen where the temporal relationships

Pregnancy tAUC	Δ	Pregnancy PPV80	Δ	MICU tAUC	Δ	MICU PPV80	Δ
NPO	0.190	Uric Acid	20	Esomeprazole	.224	Hang Blood	14
Blood Cell Profile	.164	Prenatal Vitamin	13	Weight Metric	.188	Prednisone	11
Syphilis Screen	.135	LDH	7	Glucose Beside	.169	Albuterol	7
Vicodin	.105	Fetal Ultrasound	7	Sputum Culture	.133	Na Phosphate	7
Prenatal Vitamin	-.036	IV Flush	-2	LDH	-.089	Wean Ventilator	-5
Clear Liquid Diet	-.066	Morphine Pump	-2	C. Diff	-0.122	Urinalysis	-5
Ketorolac	-.079	Epifoam	-2	Norephinephrine	-.127	Etomidate	-7
Epifoam	-.106	Ketorolac	-3	Vasopressin	-.175	Sputum Culture	-10

Table 8.4. Order name and differential (between temporal network and baseline network) for tAUC and PPV80 of the four best and worst predictions in each domain.

are the strongest. The least benefit is seen when adding temporal relationships does not significantly add to the information about choosing the order (for example, choosing to order a ventilator weaning protocol is much more complex than just 'what was ordered last?').

3.4.1. *Limitations.* This TF-TCDBN methodology unmasks temporal relationships with only a moderate increase in complexity. The principal limitation is that time is captured only approximately and only within the hospitalization. For one, the time intervals are not constant (order sessions might be separated by minutes or hours), so the level of immediacy in the temporal relationship is discarded. This could result in temporal relationships that remain hidden in the data. Second, we use a single decay factor for all variables and consider only what happens within the current hospitalization. Some test results (e.g., a blood pressure reading) may not remain relevant even beyond the current order session, whereas some diagnoses last many years. Longitudinal feature-specific decay factors is an area of future research. Third, our implementation allows only $\rho = 1$ in inference, which likely impacted performance in testing long hospitalizations.

4. Conclusions

In this chapter, we have developed a method to learn and reason about time in Bayesian networks that only moderately increases complexity and can have great benefits when a temporal relationship exists in the data.

PART 3

The System and Evaluation

CHAPTER 9

Toward a Final System

Figure 9.1. High-level system components.

The three high-level components of our overall methodology are shown in Figure 9.1.

In Part 1 of this dissertation, we primarily engaged the two boxes on the left and right. We generated deidentified data sets of three years of Gopher data in the ED, inpatient, and urgent visit clinics (the data box). Then we established methodologies for developing situation-specific treatment suggestion lists in an evolving clinical context (the inference box). We also introduced evaluation measures of those suggestion lists based on a test set of clinical data. (Programs 4.1 and 4.2). For the middle box, we utilized the existing GES and ML algorithms in the Tetrad toolkit.

In Part 2 of this dissertation, we tackled the middle box by developing novel methodologies for learning Bayesian networks in large domains involving time. We developed a principled approach for feature selection and a very fast algorithm for learning Bayesian networks. Both were based on a relational database tool we called the Maximal Association graph. We developed the Time-Forward Temporal-Causal Dynamic Bayesian Network (TF-TCDBN), a temporal Bayesian network designed to suggest actions in an observed system without major increase in complexity. We also revised our evaluation methods from Part 1 to support TF-TCDBNs (Programs 8.3 and 8.2.) With these three tools, we have the machinery to rapidly explore and evaluate domain-specific networks.

In this final part, we put the pieces together: our treatment suggestion methods from Part 1, supplemented by the new learning methodologies we developed in Part 2. In this chapter, we revisit and revise the outer two boxes in Figure 9.1. In the next chapter, we review the system we have developed and perform a larger scale evaluation, involving both retrospective data analyses and a small prospective analysis (via a survey).

1. Additional Data

The focus of this work is to suggest treatments only from data in the order-entry system: previous orders and diagnoses. Our results have shown this works rather well in some domains. However, it is also one of our principal limitations. For one, it is not the way clinicians think. For example, a clinician would want the *result* of a pregnancy test before ordering an obstetrics consult for abdominal pain, even though statistically the consult is worth suggesting if pregnancy is suspected (demonstrated by ordering the test). Also, order-entry data alone provides a limited view into the patient visit. Many data sources interact in a clinical encounter - order entry, the admission system, laboratory results, billing data, and information on previous encounters, among others.

1.1. Adding test results and demographics. As a preliminary exploration of our system in an environment of more data sources, we chose to add two additional data streams from the Indiana Network for Patient Care (INPC), which connects many healthcare-providing entities (e.g., hospital, clinics, and laboratories) across central Indiana (McDonald et al., 2005). We extracted laboratory test results and demographics, which we linked to our Gopher data sets.

Because both of these types of data are purely observations, we utilize a Markov blanket approach to choose which features to include, as opposed to our importance-based feature selection. (Recall from Chapter 6 that a target's Markov Blanket is often the best set of predictive features.) Therefore we made the following three changes to our methodologies.

- We add all of these observational features to all datasets after feature selection is completed, if they occur in our selected hospitalizations.
- We modified TEMPORALIZE-DATSET (Program 8.1) to include such features in only the $t-1$ layer. (Recall from Chapter 8 that the $t-1$ layer in a TF-TCDBN reflects past observations and not predictive targets.)
- MAG-BS is used without modification to 'choose' which of these to include by learning the graph. (Recall from Chapter 7 that MAG-BS uses a hybrid learning approach involving Markov Blanket learning to construct the graph skeleton.)

Our specific methods for each dataset is described below.

1.1.1. *Result Data.* All test results performed in participating labs (not at the bedside) with numerical results were extracted from the INPC. We encoded these as binary variables, set to true only if the result falls outside the 'critical threshold' defined by Gopher (this is meant to be far enough outside the normal range that the result is 'important').

Any results which fell between the admission and discharge date for a hospitalization were associated with that hospitalization. Some test results were associated with multiple hospitalizations, when two hospitalizations (e.g. an ED visit and an inpatient stay) appeared on the same day. We assigned the test result to the session that followed its posting to the INPC. We were only able to extract test dates (not the time), so occasionally results were associated to the session in which the test was placed. To render this somewhat innocuous, we disallowed test results from predicting other tests (via the Tetrad edge restriction API).

This amounted 39,381 abnormal (critical) test results and 1,806,419 normal test results, of 112 different types. We created a separate database of these data, which we merged on-demand with our domain-specific datasets to learn the final graph, as above.

1.1.2. *Demographic data.* We extracted gender (Male or Female), race (Caucasian, African-American, Hispanic, or Other), and age (in ten year blocks, where the final block is 90+) from the hospital admission system. These are not considered identifiers according to 45 CFR A 164.514b, so our data remained deidentified.

Two demographic features (age and race) were categorical rather than binary variables. Therefore MAG-BS could not be used, because MAG supports only binary variables. Therefore we modified MAG to fully connect all non-binary features to all others, allowing us to use MAG-BS.

Also, because demographic information is essentially eternal and does not decay in relevance, we also modified TEMPORALIZE-DATASET (Program 8.1) to use $\rho = 1$ for demographic data, regardless of the user-specified ρ.

1.1.3. *GUI changes.* To accommodate results and demographics, we added these categories to our GUI. Our evaluation tools did not require modification. This version of our GUI is shown in Figure 9.2.

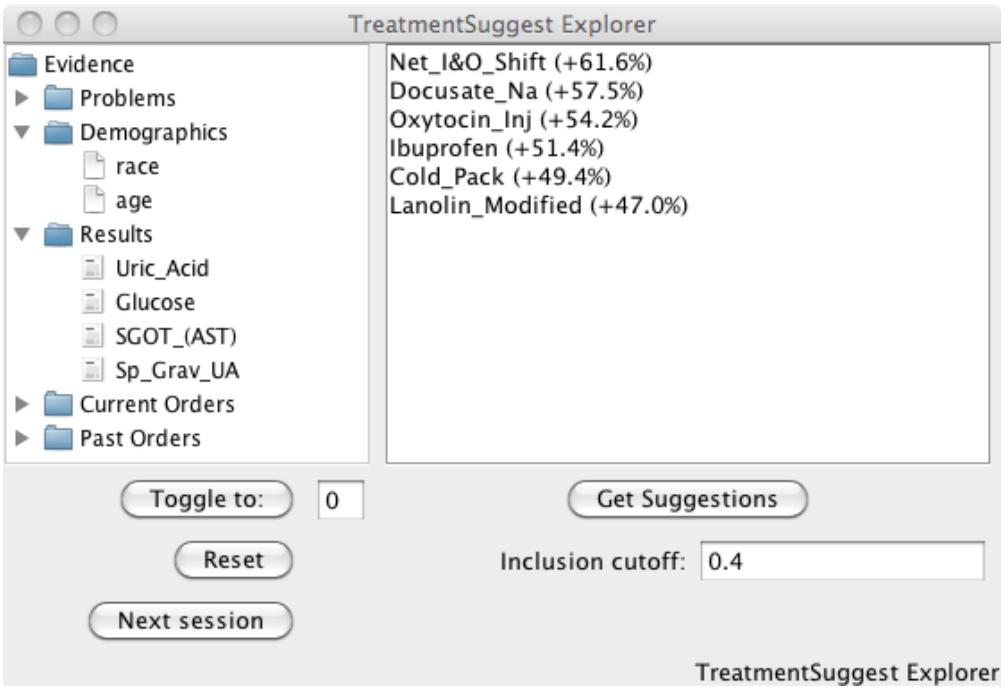


Figure 9.2. A GUI incorporating results and demographics. Shown displaying a list of available results and demographics, and a suggestion list for postpartum.

1.2. Evaluation. To study the effect of these additional data elements on our treatment suggestion menus, we learned networks of 40 orders and 10 co-occurring diagnoses using the four domains in Chapter 4 (back pain in the emergency department, hypertension in the urgent visit clinic, the medical ICU, and inpatient pregnancy). We learned four versions of each network, each using the same dataset of order-entry data and the same features. One version of the network was not augmented with additional data. In the remaining three versions, the dataset was augmented with demographic data, result data, or both, respectively. We ran Eval on these 16 networks and computed the (unweighted) average tAUC and average PPV80. These results are summarized in Figure 9.3. As a representative example of observational features selected by MAG-BS, we show results and demographics added to the emergency department network in Table 9.1.

1.2.1. *Discussion.* Although common sense would suggest that more predictors result in better predictions, the literature has shown this is not generally the case (e.g. (Kohavi and John, 1997)). Figure 9.3 plays this out. Adding either results or both results and

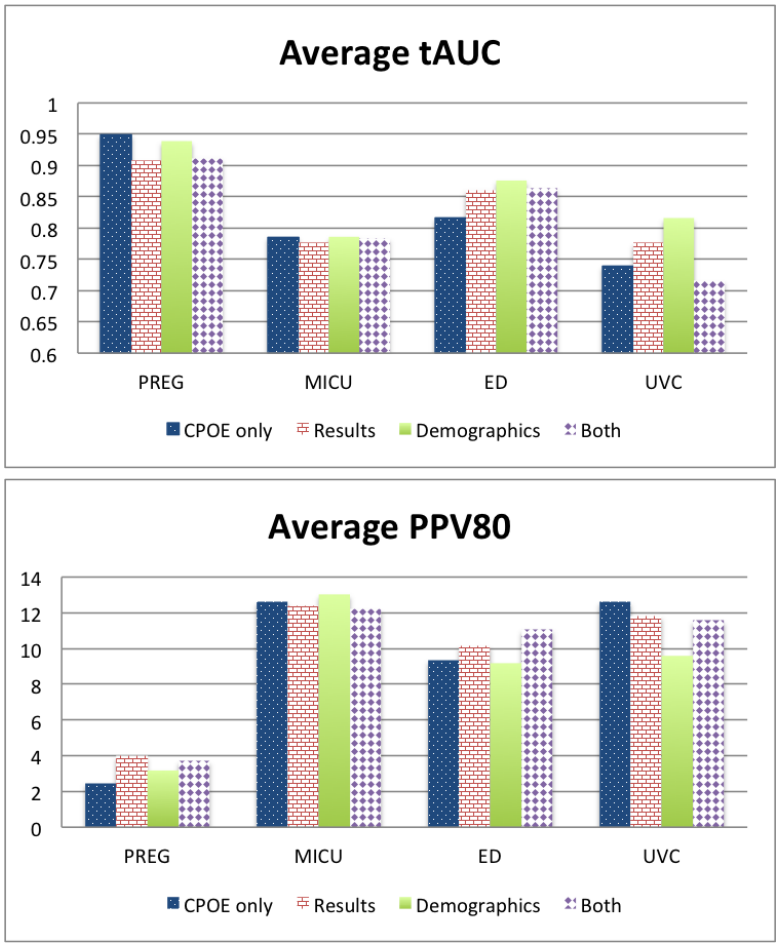


Figure 9.3. The effect of additional data on four networks of 40 orders and 10 co-occurring problems. CPOE is only these 50 features, results adds 112 types of test results, demographics adds 3 pieces of demographic inormation, and both adds all 115 features.

demographics has a negative impact on performance more often than it has a positive one. Adding only demographic features, however, has a generally positive impact. In the ED and UVC, tAUC increases noticeably (by .06 and .07, respectively). Only in pregnancy does tAUC get worse, and then only by .01. PPV80 changes were very small (much less than 1) across all sixteen networks, except in two cases. In these two cases, our earlier conclusion holds (demographics slightly improve performance, but other combinations worsen it).

Because our system learns networks which approximate the statistical relationships in the training data, the reasons for inconsistent changes in performance could be related to inconsistent statistical relationships. Figure 9.1 illustrates that this is likely the case. In

Result	→	Child Node	Explanation
Blood Urea Nitrogen	→	Abdomen CT	*nonspecific*
A1 Antitrypsin	→	Pneumonia	Low A1AC causes pulmonary problems
Phosphate Level	→	Chest X-ray	*nonspecific*
Phosphate Level	→	Lactic Acidosis	Test checks for disease
White Blood Count	→	Morphine	*uncorrelated*
Demographic	→	Child Node	Explanation
Sex	→	Pregnancy Test	Only called for in females
Sex	→	Vaginal Infection Test	Only called for in females
Race	→	Cardiac Markers	Ethnicity can impact heart disease risk
Race	→	Chest X-ray	Ethnicity can impact heart disease risk
Age	→	Lipase	Both are correlated with pancreatic disease
Age	→	Chest XRay	Age can impact heart disease risk
Age	→	Cardiac Markers	Age can impact heart disease risk
Age	→	Metabolic Panel	*nonspecific*

Table 9.1. Observational nodes that appear in the augmented back-pain-emergency-department network. The nodes and their direct children are shown, along with medical explanations for the correlation. Italics indicates no direct explanation.

three out of five result edges, there is no strong medically-relevant correlation between the nodes. This suggests that the result data is capturing data artifacts and not consistent knowledge. On the other hand, demographic edges have a medically-relevant correlation in seven of eight cases. The higher performance of adding only demographics is therefore likely related to the consistent knowledge the data encodes.

1.2.2. *Limitations and Future Directions.* Such data-artifact problems might be solved in two ways. One is to improve the learning algorithms. A possibility in BN learning might be to favor more consistent data when choosing edges. The other approach is to improve the data. For this study, we used a simple binary encoding of result data. Decision-making regarding results sometimes requires more contextual knowledge than the normal or abnormal status of a result. For example, sometimes the actual value might be less important than whether the result is trending upward or downward. Another issue is that many result dates did not fall within a recorded hospitalization and so were not included in our final data, even if they were relevant to the next hospitalization. Furthermore, only having dates of lab results (and not the time the hospital received the result) certainly resulted in erroneous temporal relationships that could have impacted performance.

1.2.3. *Conclusion.* This section highlights the difficulty of integrating disparate data sources while also showing glimpses of the benefits. In this experiment, including a small

set of fairly complete observations (i.e. demographics) improves prediction, whereas a less complete set hurts prediction, even though the information it provides seems intuitively quite relevant to the task at hand (i.e. results). This suggests that data completeness could be an important consideration when choosing what data sources to integrate.

2. Decision Thresholds

We noted in Chapter 4 that tAUC (the probability that a false classification has lower probability than a true classification) is not completely correlated with PPV80 (the length of a suggestion list that contains the next action). The problem is that TreatmentSuggest lists treatments in reverse order of posterior probability, but the posterior probability does not always predict the next action. The posterior probability might shift upward in sessions where it is selected (this is the measured by the tAUC) while still always being quite low or high due to its prior probability. This was the case with a lumbar spine X-ray in the emergency department (which always had high probability) and nalbuphine (which always had low probability).

Therefore in decision-making networks we are less interested in the probability distribution than we are in capturing these 'shifts' in probability. It is possible to monitor these shifts rather than the actual probability. One method is through decision thresholds, which are the critical probabilities for each variable, below which the action is not taken and above which it is. There is infrequently one such number, but it is possible to choose a good approximation.

2.1. Design. Because no decision threshold is perfect, we must choose a trade-off between capturing more positives (greater sensitivity) and avoiding false negatives (greater specificity). Intuitively, we want to choose the threshold that maximizes the separation between percent of correct predictions (true positives) from percent of incorrect predictions (false positives). These quantities are known as the true positive rate (tpr) and false positive rate (fpr).

A plot of the tpr vs. the fpr is a curve known as the receiver-operator characteristic (ROC curve), which we showed examples of in chapter 3. tAUC is a measure of the area

underneath this curve. Each point on the curve represents a tradeoff of accepting a percentage of false positives in exchange for capturing more true positives. An example of an ROC curve can be found in Figure 9.4.

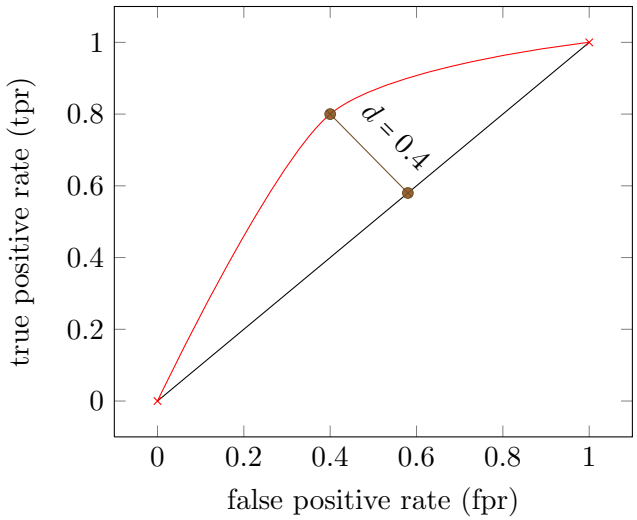


Figure 9.4. An example of finding the distance from the no-discrimination line to a point on the ROC curve, using Definition 9. Maximizing this distance is equivalent to finding an optimal point when the costs of false negatives and false positives are equal.

An optimal point can be found in the following manner. A diagonal line across the plot (from lower-left to upper-right) is the 'no-discrimination' line. The closer points are to this line, the less able the system is to distinguish true values from false values. The point furthest above the no-discrimination is the optimal point. (Note that this is only true if the cost of false positives is equal to the cost of false negatives. In other cases, we would choose the point furthest from a line of different slope, known as an *isocost* line.) (Buhmann et al., 2011) We can find such a point through simple geometry.

DEFINITION 9. *The point-line distance to a point* (m, n) *from a line* $Ax + By + C = 0$ *is*[1]*:* $d = \frac{|(Am+Bn+C)|}{\sqrt{(A^2+B^2)}}$.

Because the no-discrimination line is $-x + y = 0$, distance from a point (m, n) on the ROC curve reduces to $d = |n - m|$. We remove the $||$ because we want the greatest distance

[1]http://www.intmath.com/plane-analytic-geometry/perpendicular-distance-point-line.php

above the line. Therefore, if the cost of mistakes are uniform, the decision threshold is the probability that maximizes (tpr-fpr). This is shown in Definition 10.

DEFINITION 10. *The optimal decision threshold, if the cost of mistakes is uniform, is the probability associated with the point on the ROC curve that maximizes $tpr - fpr$, the separation between the percent of correct predictions from the percent of incorrect predictions.*

Decision thresholds suggest a more powerful type of TREATMENTSUGGEST menu. Rather than choosing an arbitrary probability cutoff (as our GUI does) or finding a list length that captures most orders (which PPV80 measures), we can list a variable-length menu of all orders that occur above the decision threshold. This would list all orders that are likely to be considered in the current session. This naturally leads to a new evaluation metric: how frequently does a decision-threshold-based menu include orders in the session they are selected? To measure situation-specific performance, we can break down the metric by active diagnoses: how frequently does a decision-threshold-based menu include orders in the session they are selected, for each possible active diagnosis in each session? We will implement a new statistic to measure this, the situation-specific positive predictive value with thresholds (ssPPV$_t$) below.

2.2. Implementation. Learning decision thresholds can be done by modifying EVAL (Program 4.2). EVAL generates vectors of probabilities for each order in each session, grouped into true instances and false instances ($\vec{T}_o$ and $\vec{F}_o$). EVAL uses these to compute the tAUC on test data, but they can be utilized to find decision thresholds on training data. For each variable, for each possible probability threshold, we count the number in each vector with at least that probability. This is the number of true positives and false positives. Dividing by the size of each vector gives us the tpr and fpr. The threshold with the largest difference is the optimal threshold according to Definition 10. This is outlined in Program 9.1.

We modified our SMILE exporter to optionally run Program 9.1 (using the training dataset) before the final export, embedding the threshold value into the exported network. We also modified both TREATMENTSUGGEST and EVAL to sort the probability lists by the

difference between probability and threshold, if a threshold is present. We further modified our GUI to list all treatments in the suggestion pane that are above the decision threshold (rather than a probability cutoff). The GUI changes are shown in Figure 9.5.

Finally, we modified EVAL to compute the $ssPPV_t$ (situation-specific positive predictive value using thresholds, described in the prior section). This involves computing a vector $\vec{D}_o$ for each diagnosis in the network. Each vector contains the counts of each order that appears above the decision threshold in the session where it was made, for all sessions in which that diagnosis was active. Dividing these by the total number of each order where the diagnosis is active gives the positive predictive value.

2.3. Evaluation. We sought to test whether decision thresholds result in relevant orders appearing earlier in suggestion lists. To test this, we computed PPV80 values in the temporal MICU and pregnancy networks from Chapter 8 both before and after running

Program 9.1 Find-thresholds(G,D)

Input: G is a Bayesian Network Model
Input: D is a training dataset of hospitalizations, consisting of orders and co-occurring problems by session.
thresholds[] is an array of optimal thresholds for each order
Run Eval(G,D) and get $\vec{T}_o$, $\vec{F}_o$.
for all $o \in G$ **do**
  score[] is an array of scores for each possible probability threshold
  for $i = 0 \rightarrow 100$ **do**
    tp = 0
    fp = 0
    for all $p \in \vec{T}_o$ **do**
      if $p \geq i$ **then**
        $tp = tp + 1$
      end if
    end for
    for all $p \in \vec{F}_o$ **do**
      if $f \geq i$ **then**
        $fp = fp + 1$
      end if
    end for
    score$[i] = \frac{tp}{\|\vec{T}_o\|} - \frac{fp}{\|\vec{f}_o\|}$
  end for
  thresholds$[o] = \underset{i}{\text{argmax}}\ \text{score}[i]$
end for
Return the array of thresholds

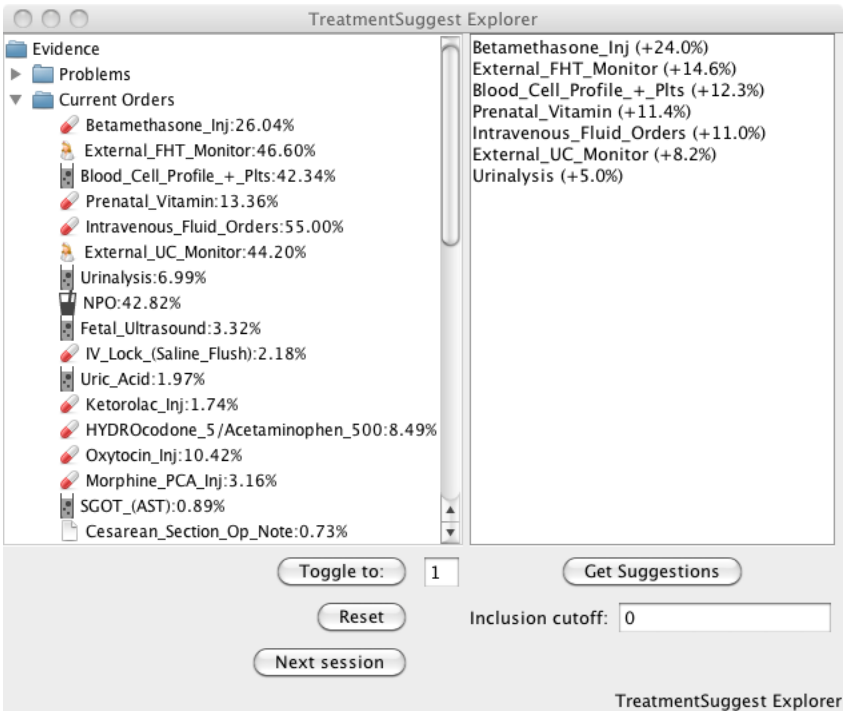


Figure 9.5. A modified GUI incorporating decision thresholds. Shown displaying a network for pregnancy with preterm labor selected as evidence. The left pane demonstrates that sorting by difference from the decision threshold produces a remarkably different ordering than the posterior probability distribution. The right panel shows a treatment suggestion list of all orders above the decision threshold.

Program 9.1 on the training set. (tAUC is unchanged by decision thresholds and ssPPV$_t$ is only computable in networks with thresholds, so these were not included in our comparison.) Average PPV80 and a histogram of the two sets of PPV80 values are shown in Figure 9.6. The orders with the largest shifts (for better and worse) are shown in Table 9.2.

2.3.1. *Discussion.* This change showed only moderate overall improvements to the *average* length of our list in pregnancy and MICU networks (average Δ PPV80 .85 and 1.84 respectively).

In the pregnancy network, however, the *maximum* list length was quite reduced. The largest PPV80 without thresholds was 25 (for Fetal Ultrasound) but only 10 with thresholds (for IV Fluids). The histogram also demonstrates that the items previously in the tail are

Network	Pregnancy avg/max PPV80		MICU avg/max PPV80	
No thresholds	4.025	25	15.47	36
Thresholds	3.175	10	13.63	34

(a) Average PPV80.

Pregnancy PPV80 Δ		MICU PPV80 Δ	
Fetal Ultrasound	18	Phys. Therapy Consult	18
Urinalysis	15	Urinalysis	16
Betamethasone Inj	13	Pharmacokinetics Consult	12
Uric Acid	-4	Vasopressin	16
Oxytocin Inj	-5	Nutrition Consult	22
SGPT (ALT)	-5	BNP	24

(b) PPV80 with greatest Δ.

Table 9.2. (Top) Average PPV80 using Chapter 8's pregnancy and MICU networks, with and without thresholds. (Bottom) In the same networks, the orders with largest difference in PPV80 (the amount smaller using a threshold compared to no threshold).

now closer to the center. The orders most affected tended to be rarer and not general-purpose. Two of the top three improved PPV80s are used primarily with preterm labor (betamethasone and a fetal ultrasound).

Although the average PPV80 difference was larger in the MICU network, specific improvements were harder to discern. The largest PPV80 without thresholds was 36 (for Succinylcholine) but still 34 with thresholds (for liver tests). The histogram shows a reduction in orders in the tail, but not as dramatic as in pregnancy. The most improved PPV80s were, while not extremely general, also not extremely specific (e.g., physical therapy consult).

From a statistical standpoint, adding decision thresholds resulted in moderate overall positive impact on PPV80 length. On closer inspection, decision thresholds might hold significant clinical value in the pregnancy domain, because they allow correct prediction of rarer orders with specialized use. This was not the case in the MICU, where the types of affected orders was not as discernible. However, this could be because all orders in the MICU are rare, as our test set only involves a few hundred hospitalizations.

Overall, we believe that decision thresholds make moderate statistical improvements and can have significant clinical impact (in that correct rarer orders are suggested more accurately) when there is a distribution of rarity and sufficient data.

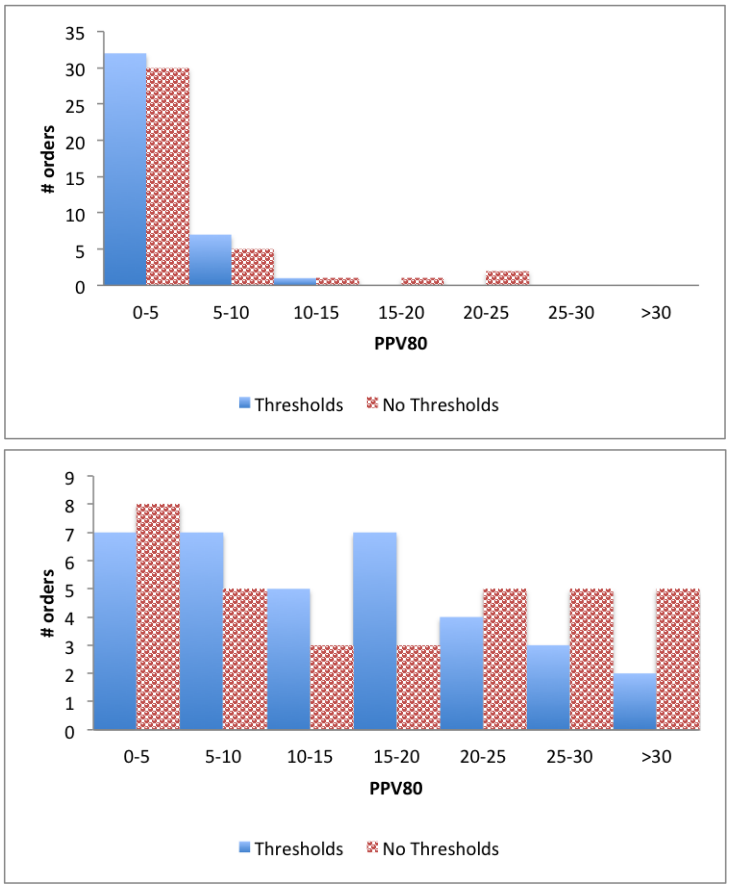


Figure 9.6. Histogram of PPV80 distribution with and without thresholds. Top is the pregnancy network, bottom is MICU.

CHAPTER 10

Evaluations on Inpatient Medicine

In the previous chapters, we developed: Gopher datasets, optionally augmented with demographic and result data; a tool to provide situation-specific treatment suggestions, with and without decision thresholds; three evaluation measures of those suggestions (tAUC, PPV80, and PPV_t); a SQL tool to compute the Maximal Association Graph (MAG); a feature selection program based on association rule mining and graph-theoretic importance in the MAG (FS-Important); a fast Bayesian network structure learner based on the MAG, Markov Blankets, and a greedy hill-climbing search (MAG-BS); and a methodology for learning and performing inference using temporal relationships in the graph.

In this chapter we evaluate our final suite of methodologies on inpatient medicine, our primary target. First, we review our final methodology. Second, we perform an evaluation using networks for 13 common chief complaints at Wishard hospital, evaluated against our data-analytic metrics. Finally we perform a prospective evaluation of the system's accuracy and completeness in generating admission order sets, using a survey instrument.

1. Final Methodology

Throughout this work, our primary objective has been to develop a methodology to produce situation-specific treatment advice from order-entry data. In Part 1, we developed a preliminary system to do this. From hospitalization order-entry data, we induce domain-specific BNs that represent the probabilistic relationships among orders. Then, as specific orders are placed in a specific case, we instantiate the variables corresponding to those orders in the network (known as *evidence*), which revises the probabilities for other orders in the BN to the posterior probability that they would be placed conditioned on the previous orders, and thus allows us to rank remaining orders by their probability of occurring. In our interface, we present these to the user as orders are placed, in descending order of probability. To limit the length of such suggested order lists, we optionally also impose a cutoff to present only the most likely orders. In Part 2, we develop new methodologies that can build a BN from data at high-speed using principled feature selection and temporal reasoning. We adapt our retrospective data analysis and GUI accordingly. In the previous chapter, we augment our networks with demographic data and decision thresholds, and we list suggestions that

fall above the decision thresholds rather than those above an arbitrary probability. Here we review the interaction of our methodologies and present it as a comprehensive system.

In Chapter 1, we extracted datasets from Gopher for four modalities of medicine: GopherInpatient, GopherED, GopherUVC, and GopherWVC. These feed into a program that selects all but the irrelevant features, FEATURE-ASSOC (Program 6.1) given a set of T initial targets that define the domain. Then a dataset containing only the relevant hospitalizations and features is created, the MAG is generated (Program 5.1), and Markov Importance is used to select the most important t_0 actions and t_1 non-actions in the resulting graph (Program 6.2). Next, a per-session dataset is created using the selected features. This is optionally augmented with demographics and test results (see Chapter 9) and is converted into a TF-TCDBN dataset via TEMPORALIZE-DATASET (Program 8.1), for which a MAG is again generated and BLANKET-STITCH (Program 7.5, augmented to learn about time and categorical variables) is used to learn the graph. Decision thresholds are then learned via FIND-THRESHOLDS (Program 9.1). The final graph is converted to SMILE format.

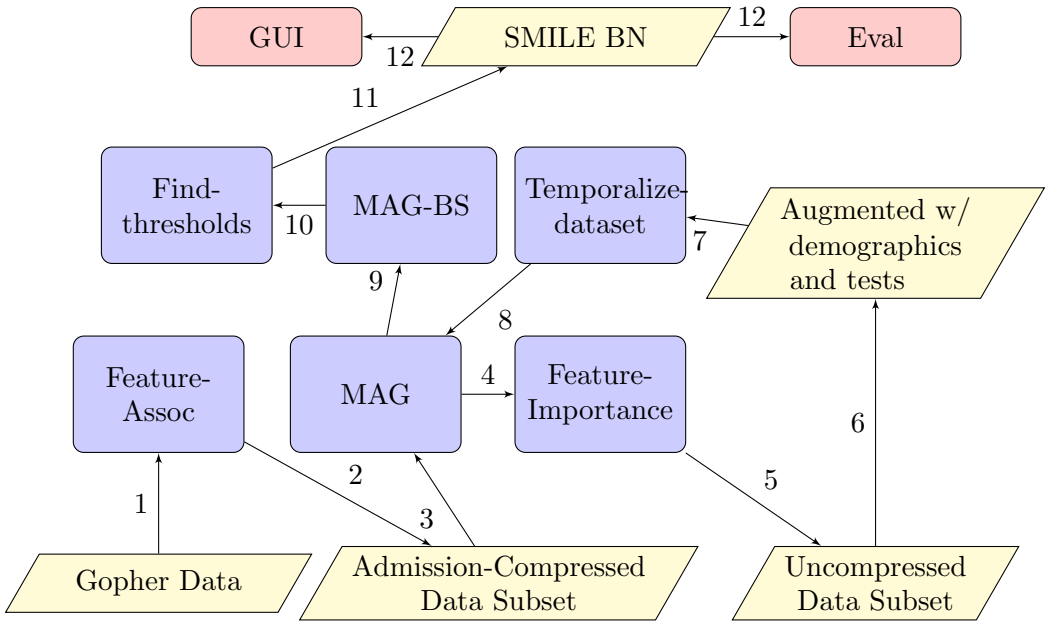


Figure 10.1. A flowchart of the components in our final system. The flow through the components is indicated by the numbered edges, which also correspond to the line numbers in Program 10.1.

These graphs can then be explored through our TreatmentSuggest GUI or evaluated on retrospective data with Eval (Program 4.2, augmented for temporal reasoning and decision thresholds). Eval computes three metrics for each order in the network:

- tAUC: The probability that the network can distinguish a true instance from a false instance, as measured by $P(P(T) > P(F))$.
- PPV_t: The probability that the network can distinguish a true instance by a specific decision threshold, as measured by $P(P(T) \geq d_v)$, where d_v is a threshold value for each feature. The ssPPV_t is a variant that evaluates the PPV_t only when specific diagnoses are active.
- PPV80: The length of the suggestion list that is needed to capture 80% of decisions.

These measures are all interrelated. The tAUC is an application-neutral measure of the predictive ability of the network. PPV_t measures the ability of a single decision threshold to separate true and false positives, and is used in our final system to produce suggestion lists. PPV80 measures how deep in the list a selected order will be found 80% of the time, which can be used to find the maximum length needed for a fixed-size suggestion list.

We compute averages for each measure. In Chapter 4, we computed average tAUC and PPV80 weighted by order prevalence, to emphasize the performance of our system in the average case. Since then, we have reported unweighted tAUC and PPV80, to show performance across all orders. With the methodological improvements in the preceding chapters, we have found unweighted averages now correlate quite well with the weighted averages. In this chapter, we compute averages for ssPPV_t (situation-specific PPV_t). We include orders with at least 20% prevalence in the presence of each specific diagnosis. This measures average performance but in specific situations, ignoring outliers in a manner similar to PPV80, without weighting by overall prevalence as in Chapter 4. To compute a whole-network average ssPPV_t, we average the averages for individual diagnoses, weighted by prevalence of the diagnosis.

Various parameters must be specified to these programs. Some are chosen through experimentation: the decay factor in Temporalize-dataset, which we set to $\rho = .8$; the confidence threshold in Find-thresholds, which we set to $\epsilon = .01$; the max-treewidth in

GREEDY-SEARCH (Program 3.1), which we set to 5; and the number of steps in Markov Importance, which we set to max(5, TW(G)).

All programs in this work were developed in SQL and Java, using Tetrad's Bayesian-learning tools and SMILE's Bayesian-inference API. We discussed our reasons for choosing these tools for temporal reasoning in Chapter 8. Some further justification for Tetrad and SMILE over other Bayesian network reasoning tools can be found in Appendix A.

Together, these components form a comprehensive system for quickly generating and evaluating domain-specific networks in inpatient medicine. A diagram of the components involved can be seen in Figure 10.1. We wrote a program in Java to connect all of these components, Program 10.1, the line numbers of which correspond to the edges in the diagram.

Program 10.1 Learn(D,T,n_0,n_1,n_{1p})

Input: D is a a training dataset

Input: T, n_0, n_1 are a set of targets and number of actions and non-actions as defined in Program 6.2.

1: V=FEATURE-ASSOC(T,D,n_0,n_1). {V is an output set of features from a source dataset D and T is a list of domain-defining targets.}
2: $D_v \subset_V D$ {D_f is a subset of D containing all the features in V, with one row per admission.}
3: M_v=MAG(D_v)
4: F=FEATURE-IMPORTANCE(D,V,n_0,n_1,max(5, TW(M_v)) {Chooses n_0 actions and n_1 using the previously generated MAG.}
5: D_0 and D_1 become training/testing subsets of D containing features from $F \cap V$, with one row per session, split into 2/3 training data and 1/3 test data.
6: D_0 and D_1 are optionally augmented with demographic and/or test result information (using up to 115 features). By default, demographics are included and test results are not.
7: D_T=TEMPORALIZE-DATASET(D_0,I_0,.8) {D_T becomes a temporal dataset appropriate for learning a TF-TCDBN.}
8: M=MAG(D_T). {Learn the MAG on the temporal dataset, fully connecting any non-binary variables.}
9: G=BLANKET-STITCH(D_T,M) {Learn the network using the temporal dataset with the MAG as a restriction graph.}
10: G_t=FIND-THRESHOLDS(G,D_0) {Find decision thresholds in the graph using the training data.}
11: Save G_t to a SMILE-format Bayesian network using our converter.
12: Evaluate either with EVAL(G_t,D_1) or with TREATMENTSUGGESTINTERACTIVETEMPORAL(G_t)

1.1. Comparison to Chapter 4. To compare the final system to the overall results in Chapter 4, we ran LEARN(D,T,40,10) four times, where D and T were the four modalities and targets defined in Chapter 4. Table 10.1 reports the unweighted average tAUC and PPV80 alongside those computed by the methodology in Chapter 4.

	Inpatient Pregnancy		ED Back Pain		MICU		UVC Hypertension	
	Ch4	**Ch10**	**Ch4**	**Ch10**	**Ch4**	**Ch10**	**Ch4**	**Ch10**
tAUC	0.791	0.935	0.779	0.868	0.689	0.808	0.741	0.798
PPV80	6.47	4.40	14.05	12.32	16.23	13.08	13.19	8.38

Table 10.1. Unweighted average tAUC and PPV80 for the four domains defined in Chapter 4, using that methodology ('Ch4') compared to Program 10.1 ('Ch10').

1.1.1. *Discussion.* Table 10.1 shows significant average improvement in both measures across all networks. The average tAUC improved .10, and the average PPV80 shrunk by 2.94. This analysis does not show the greatly-decreased learning time, the more relevant set of features selected, the temporal relationships captured by the new networks, or the improved performance for particular orders. Those improvements have been evaluated in the preceding chapters. This table does, however, provide a snapshot of the improvements made.

2. Partitioned Networks in Inpatient Medicine

As discussed in Chapter 5, it is not possible to learn a single Bayesian network for all of medicine. However, it is possible to create a large set of domain-specific networks that can be heuristically chosen on demand. Heckerman and Nathwani's (1992) Pathfinder utilized this approach, calling these subnetworks *partitioned networks*, where each partition consisted of a subnetwork which had very few interdependencies with the other subnetworks. In this way, they could reason on these subnetworks rather than the full network. We will adopt this approach in inpatient hospitalization.

Each inpatient hospitalization has exactly one chief complaint. We reasoned that by learning domain-specific networks for chief complaints, we could effectively develop a set of networks to provide treatment suggestions given any chief complaint. Using this intuition,

we created 12 subnetworks using chief complaints from GopherInpatient. Each subnetwork included one primary chief complaint and all highly-related secondary chief complaints. By 'highly related,' we mean any chief complaint which also frequently appears as a secondary problem alongside the primary complaint. We chose the three primary complaints that were the most frequent in our data, and then asked an independent physician rater to choose nine more. The physician was given a list of the top 100 chief complaints in GopherInpatient and asked to chose those he considered most important and most directly treatable. The chief complaints for each of the twelve networks are listed in Table 10.2.

2.1. Methods. We ran LEARN(GOPHERINPAT,T,80,20) 12 times, with T equal to each list of complaints in Table 10.2. A conglomerate view of these networks can be seen in Figure 10.2. We then used EVAL to calculate tAUC, PPV80, and ssPPV$_t$ and averages, as described in Section 10.1.

We do not explicitly measure performance at different times within a hospitalization, but the ssPPV$_t$ implicitly captures temporal performance when a diagnosis sequence implies a temporal ordering. For example, one can intuit the temporal sequence pregnant→labor→postpartum, and therefore one could examine the ssPPV$_t$ value for these three diagnoses to gain some insight about how the system performs at these three stages of the hospitalization.

Primary Complaint	**Secondary Complaint(s)**
Pregnancy	Labor, Postpartum
Normal Term Infant	Normal Newborn
Chronic Renal Failure	Acute Renal Failure, Chronic Renal Insufficiency
Syncope	Dizziness
GI Bleed	Rectal Bleed, Hematemesis
CHF	Chest Pain
Stroke	TIA
Cellulitis	Abscess
Unstable Angina	Angina Pectoris, Recurrent Angina
Abdominal Pain	Small Bowel Obstruction, Cholecystitis, Acute Pancreatitis
Altered Mental State	Drug Overdose
Pneumonia	Cough

Table 10.2. The chief complaints targeted by each of the twelve partitioned networks of inpatient medicine. These include: a primary complaint – the three most frequent (in italics) and nine additional complaints judged by a physician to be the most important and directly treatable among top 100; and secondary complaints – any chief complaints which frequently occurred as secondary problems amidst the primary complaints.

2.2. Results. Program 10.1 (LEARN) ran in an average of 50 minutes per network, not including time to compute evaluation statistics. The maximum runtime was 105 minutes for syncope, in which the slowest step was performing inference on the training data to compute thresholds. FIND-THRESHOLDS was the most widely varying factor in learning time; each network took from 10ms to 1s per hospitalization for inference.

Summary statistics for all 12 networks are shown in Figure 10.3, which are shown graphically in Figure 10.3. The ssPPV$_t$ for each diagnosis in each of the 12 networks are shown in Table 10.5. Figure 10.4 summarizes this through a bar chart of average ssPPV$_t$ with variance (error bars). A red dash denotes the average ssPPV$_t$ for the network-defining diagnoses, from Table 10.2. The individual orders in all situation-specific PPV$_t$ calculations in all networks are too vast to show, but a small representative sample are shown in Table 10.4. Note that these are not the suggestion menu themselves, but the frequency with which each highly-correlated order appears in the suggestion menu when the listed diagnosis is active.

2.3. Discussion. The system has strong overall predictive performance, as shown in Table 10.3. Average tAUC is ≥ 0.8 for all networks except syncope and GI bleed. The

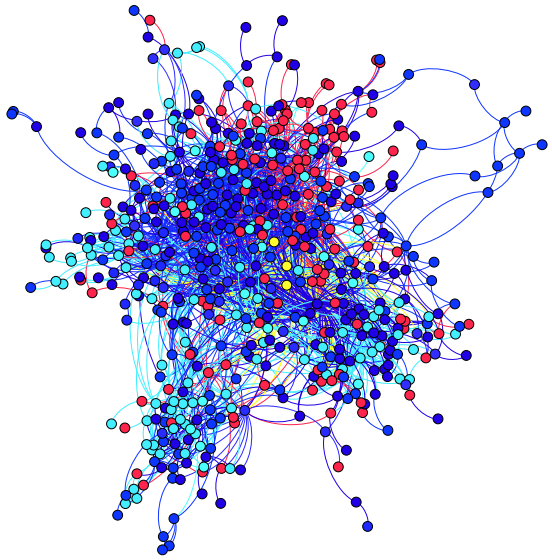


Figure 10.2. A conglomerate view of the 12 partitioned networks. Yellow nodes are demographics, red are diagnoses and complaints, light blue are nursing and diet orders, dark blue are all other orders.

PPV80 shows the resulting treatment suggestion menus contain the desired next order at menu lengths between 12%-32% of the total number of orders in the network (average 25%).

	Pneumonia	Pregnancy	Syncope	GI Bleed	Unstable Angina	Chronic Renal Failure
ssPPVt	0.831	0.934	0.863	0.796	0.882	0.882
tAUC	0.852	0.907	0.720	0.762	0.805	0.851
PPV80	18.63	10.00	17.81	21.09	21.91	22.46
	Cellulitis	Altered Mental State	Abdominal Pain	CHF	Normal Term Infant	Stroke
ssPPVt	0.798	0.819	0.788	0.839	0.988	0.864
tAUC	0.808	0.839	0.858	0.844	0.905	0.820
PPV80	23.36	21.29	19.13	19.80	10.63	24.23

Table 10.3. Summary statistics for each of the 12 networks. ssPPVt is the average situation-specific positive predictive value for all orders with at least 20% prevalence in each situation, weighted by prevalence of the situation (higher is better). tAUC is the average time-specific predictive power (higher is better). PPV80 is the average menu length to capture 80% of orders (lower is better).

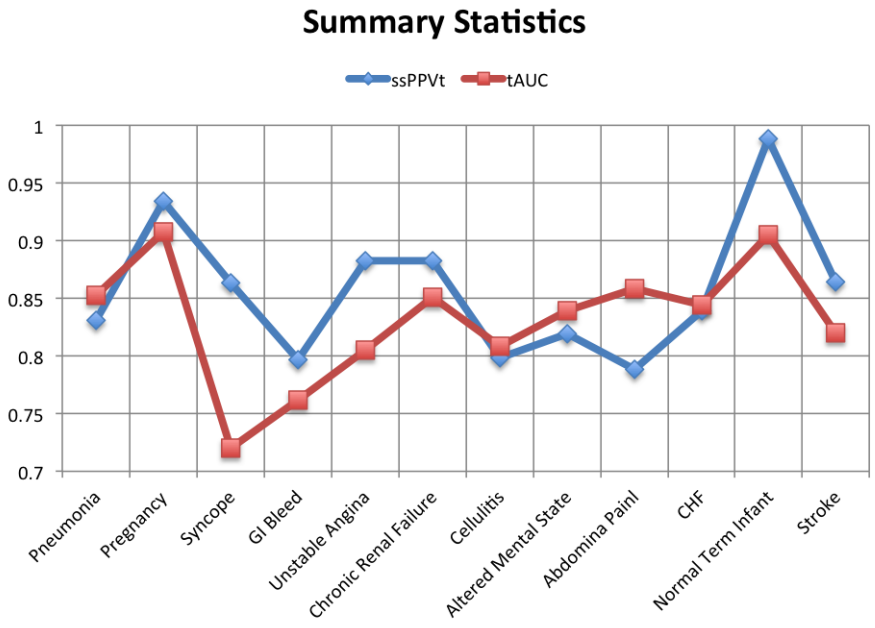


Figure 10.3. Average tAUC and ssPPVt for each of the 12 inpatient networks, from Table 10.3.

Pregnant Postpartum		Normal Infant Hyperbilirubinemia		CHF Pneumonia		Abdominal Pain Small Bowel Obstruct.	
Ob Consult	97.1%	Bili. Lights	93.5%	Metabolic Panel	85.0%	IV Fluids	94.6%
Regular Diet	89.1%	Bili. Blanket	96.0%	Cardiac Markers	73.3%	Metabolic Panel	84.4%
Ibuprofen	98.9%	Bili. Total	88.2%	Blood Culture	78.6%	NPO	95.2%
Docusate Na	97.9%	Ped. Consult	83.3%	Glucose Bedside	92.3%	Seq. Compression	82.6%
Net IO Shift	97.9%			Oxygen Therapy	58.3%	NG Tube	81.8%
Epifoam	91.4%			Ceftriaxone Inj	83.3%	Magnesium Level	96.9%
Vicodin	84.5%			Heart Healthy	83.3%	Foley Catheter	71.9%
Simethicone	90.1%			Net IO Shift	100.0%	Phosphate	100.0%
Lanolin	94.6%			Urinalysis	83.3%	Urinalysis	82.1%
Cold Pack	99.2%			Simvastatin	100.0%	Metabolic Panel	81.5%
Sitz Bath	98.0%			Protime	72.7%	Surgery Consult	96.3%
Tylenol	93.8%			Occ. Therapy	90.0%	Naloxone Inj	100.0%
Oxytocin	97.9%			Magnesium Level	100.0%	Urine Culture	96.0%
				Vancomycin	80.0%	PCA Nursing	100.0%
						Glucose Bedside	73.9%
Average	**94.7%**	**Average**	**93.2%**	**Average**	**87.1%**	**Average**	**88.2%**

Table 10.4. The PPV_t for several of the most-frequent orders in four specific situations. The first row specifies the network used; the second row is a comorbid condition within the network. The average reflects the entire set of treatments, not just those shown.

This reflects that the TreatmentSuggest methodology is on average 75% more specific than feature selection alone.

Better still is the situation-specific PPV_t, also shown in Table 10.3. Given a specific comorbid diagnosis, the desired next treatment is listed on a decision-threshold-based treatment menu on average 85% of the time. $ssPPV_t$ for diagnoses in two networks (pregnancy and normal term infant) always average above 95%, and the average is at least 95% for at least one diagnosis in each network (see Figure 10.4). Table 10.5 breaks this out in more detail. Examples of $ssPPV_t$ above 95% include abdominal pain with liver cirrhosis, syncope with chest pain, and chronic renal failure with coronary artery disease. Less than 10% of comorbid diagnoses had $ssPPV_t$ below 80%.

Figure 10.3 shows that average $ssPPV_t$ parallels average tAUC but frequently exceeds it. This is somewhat remarkable; tAUC is an application-neutral comparison of probabilities of potential predictive power, but $ssPPV_t$ is an application-specific comparison of posterior probability to a predetermined decision threshold. What this graph demonstrates is that the application-specific methodology exceeds general predictive power *by focusing on specific*

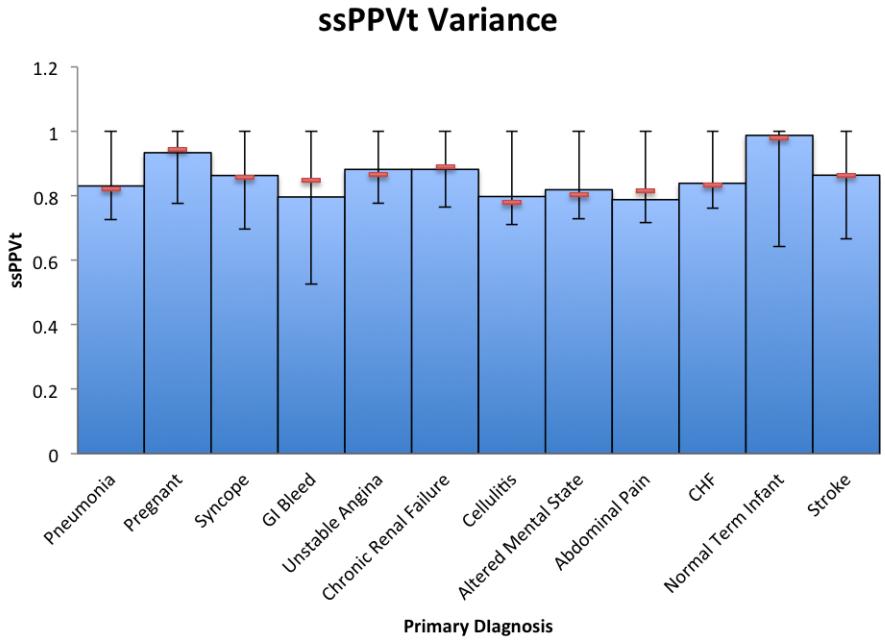


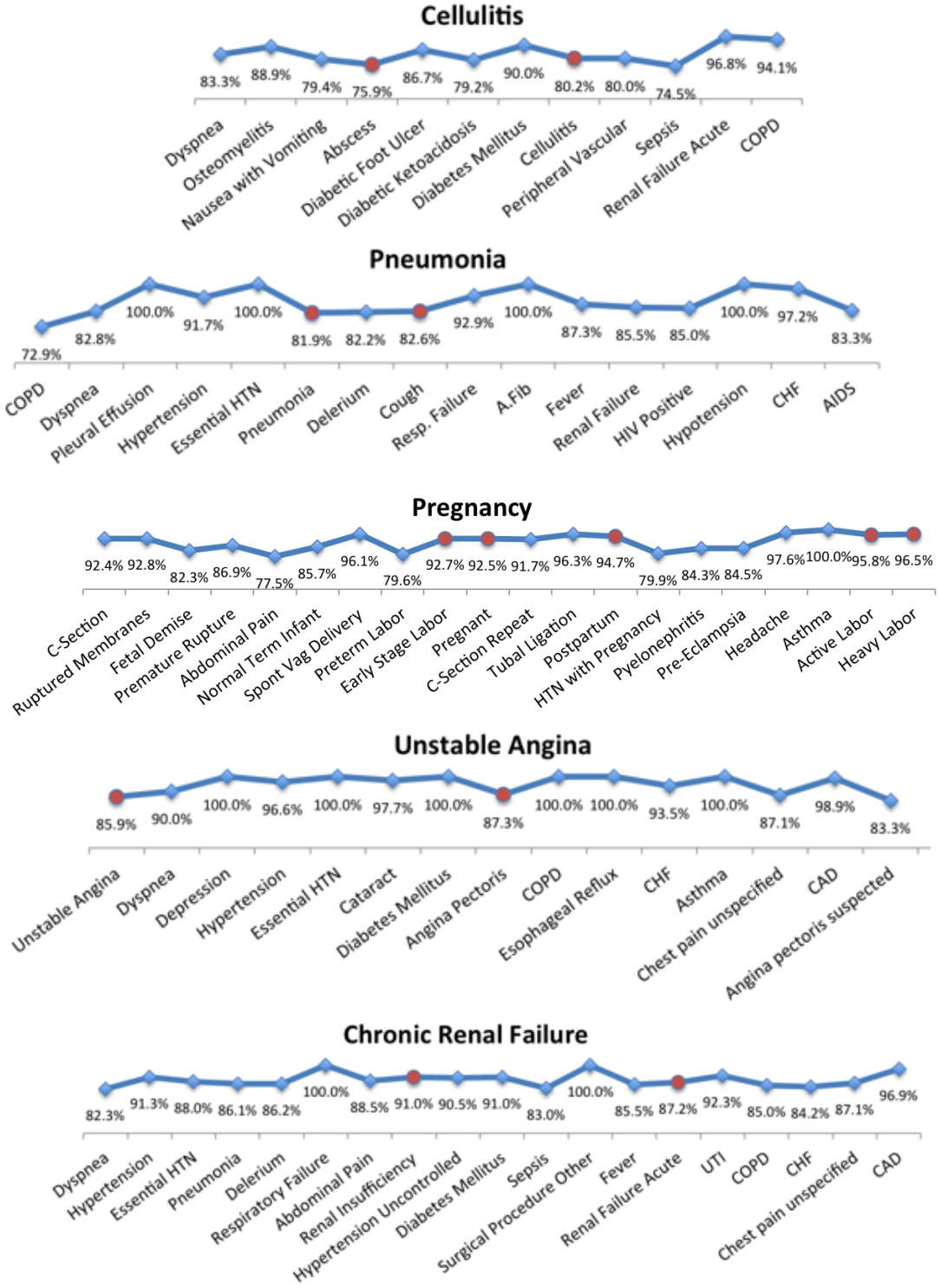
Figure 10.4. The variance of situation-specific PPV_t. The bar is the average situation-specific PPV_t, and the error bars represent the minimum and maximum $ssPPV_t$ for all diagnoses in the network. The red horizontal line shows the average $ssPPV_t$ for the network-defining diagnoses (Table 10.2)

.

situations. Average $ssPPV_t$ always involves at least one diagnosis as evidence and compares only orders consistently used when that diagnosis is present.

All this leads us to conclude we have developed a fast, comprehensive system that will suggest the most relevant treatments with high accuracy in reasonably short menus. Our system takes advantage of the ability to instantiate evidence in networks of inference and performs better as evidence increases. It outperforms its own overall predictive power when at least one diagnosis is present. The result is a system to generate partitioned networks and suggest treatments in medicine using only CPOE data. This evaluation shows the system's strong performance on inpatient medicine, which is the primary target of this work.

PPVt of specific diagnoses in each partitioned network



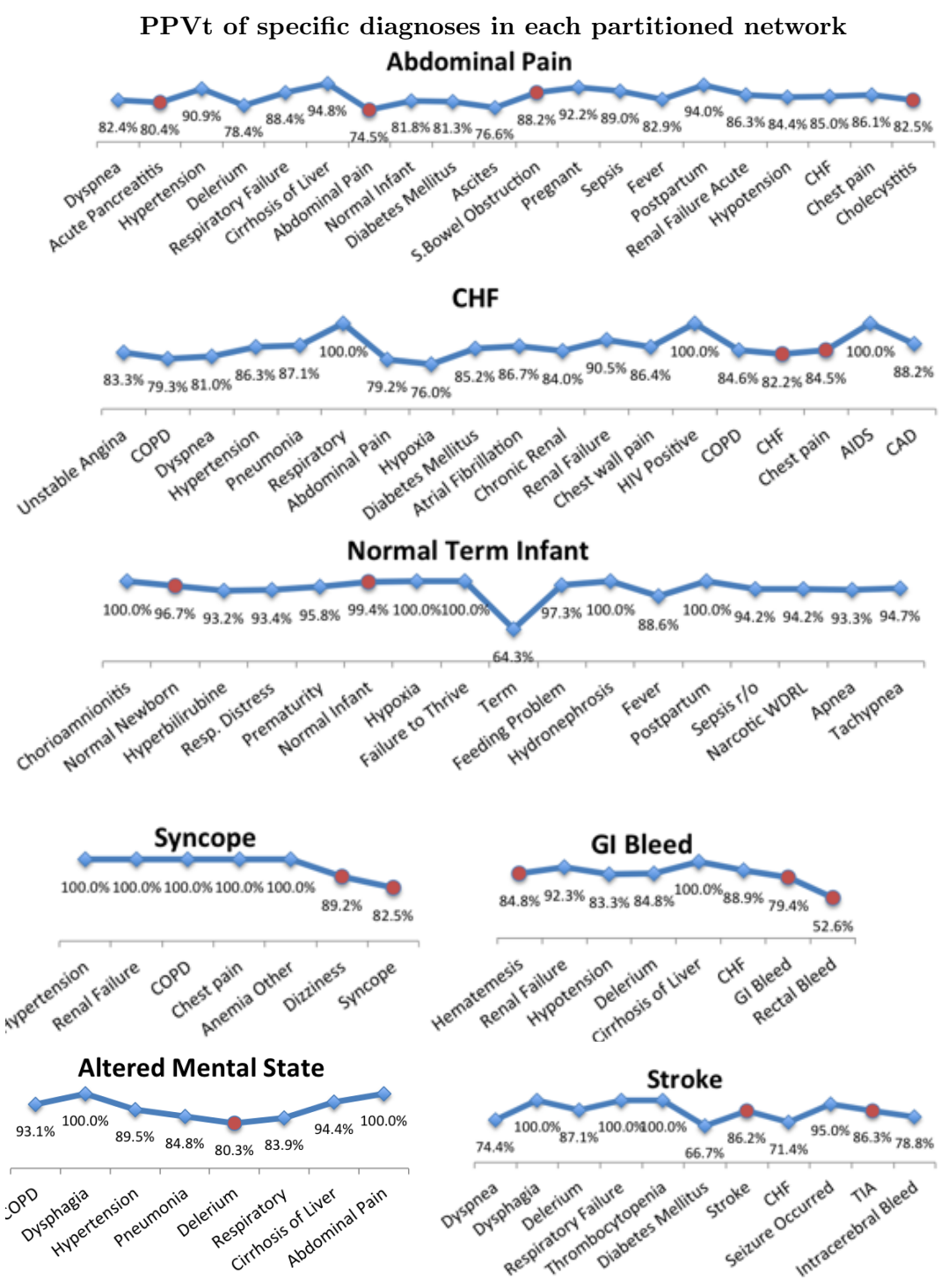


Table 10.5. Situation-specific PPV_t in each network. PPV_t is calculated using all orders used at least 20% of the time when these diagnoses are active. The red, circular points are the domain-defining diagnoses from Table 10.2.

3. Prospective evaluation: a survey

Retrospective simulation allows measurement of treatment suggestion list accuracy and completeness in arbitrarily complex situations. However, to understand whether the suggestion lists actually improve the speed and quality of healthcare delivery requires the prospective input of healthcare professionals.

We posed the hypothesis that our treatment suggestion system could generate a draft admission order set which could approximate an admission order set designed by a physician. We further hypothesized that physicians given this draft would more quickly create more complete order sets than those without the draft. To test this hypothesis, we designed a survey to study the content and speed of physicians generating order sets with and without the drafts, as well as physicians' perceptions of the drafts.

3.1. Methods. We first chose four inpatient chief complaints to survey. To choose these, we asked a physician (not included in the survey) to select four chief complaints for which he felt he could design an admission order set without assistance from external resources. The physician was asked to select from a list of the 100 most frequent chief complaints in the GopherInpatient data.

3.2. Preparation of draft order sets. We optimized decision thresholds for order set generation. EVAL (and therefore decision thresholds) record the predicted probability that each order was placed *at some point* in the order session. Because the probability frequently increases as other orders in the session are placed, such a threshold is not designed to show treatments that might be considered when nothing has been ordered. In admission order sets, however, we want a threshold that reflects whether the session might include the order *when no other orders have been placed*. Therefore we modified EVAL to optionally compute this threshold. This required a simple change. In EVAL, when evidence is set for all orders and co-occurring problems that have occurred up to this point in the hospitalization, the current session is included. The modified EVAL does not include other orders in the current session as evidence. This is used to compute decision thresholds that better reflect the beginning of an encounter.

Next, we ran LEARN(GOPHERINPAT,T,95,10) four times with these optimizations, with T set to each of the target chief complaints. In order to focus the system on the chief

complaint, we set $n_{1p} = 1$ (the number of pursued actions among the selected actions), as in the implementation of FS-IMPORTANT in Chapter 6. This allows more actions to be chosen for the single chief complaint without increasing the number of features in the network.

Finally, we used our GUI to generate a threshold-based suggestion list with only the target complaint set to true. We organized these by Gopher order type (test, radiology, drug, consult, and nursing). These four order sets can be seen in Table 10.6.

3.3. Survey recruitment and design. We recruited seven Wishard hospitalists to complete an anonymous survey. Five were full-time internal medicine physicians. One was a half-time internal medicine physician and one was a clinical nurse specialist. Among the physicians, three had more than five years of post-residency experience.

Each was asked to sit with us and write generic admission order sets for each of the four complaints. The order in which the chief complaints were presented was randomized. For two of the order sets (chosen randomly), the physician was given an order set draft generated using the method described above. We timed the physicians as they wrote each draft. They were instructed not to rush, but that they would not be allowed to revisit an earlier order set once they completed it. Throughout, we asked for qualitative comments.

Most raters did not use the order set drafts directly, writing their final order sets on separate paper and only marking notes on the draft. They frequently organized their order sets using some variant on the acronym ADCVANDALISM[1] (admit, diagnosis, condition, vitals, activity, nursing, diet, allergies, labs, intravenous fluids, special, and medications). We mapped their hand-written order sets to our draft lists in order to compute agreement with our drafts. Many sections were directly mappable (nursing, diet, allergies, labs, intravenous fluids, special, and medications). The admission section we mapped to any orders which are automatically added in that admission location (though most raters also included these explicitly). These included: bedside telemetry in all ICUs and a standing order for oxygen therapy in the PICU. Condition, vitals, and activity were ignored as they were out of scope of our order sets.

This survey was approved by the IRB (#EX1105-03).

[1] http://www.md2b.net/resources/survivalguide/surgery/postop.html

CHF

DIAGNOSTIC

3.10%	Cardiac Markers
3.10%	Blood Cell Profile + Plts
2.90%	Basic Metabolic
2.30%	Lipid Profile
1.50%	Magnesium Level
1.30%	Phos
1.00%	Urine Culture
0.70%	Cardiac Echo
0.60%	Urinalysis
0.10%	Blood Culture
0.00%	Iron/TIBC

DRUG

3.10%	Lisinopril
1.60%	Nitroglycerin Oint
1.40%	Furosemide
1.40%	Humulin NPH Insulin
1.10%	Supplemental Hum Reg Insulin
0.80%	Albuterol Nebulizer
0.30%	HydrALAZINE
0.30%	Intravenous Fluid Orders
0.10%	Morphine
0.00%	Metoprolol Succinate XL
0.00%	Clopidogrel

CONSULTS

0.60%	ACE Team Consult
0.20%	Renal Consult/Appt
0.10%	Occupational Therapy Consult
0.00%	Physical Therapy Consult

NURSING

2.60%	Bedside Telemetry
1.90%	Net I&O Shift
1.20%	Glucose Bedside
1.00%	TED hose
0.30%	Oxygen Therapy

DIET

7.70%	Low Sodium Diet
6.80%	Fluid Restriction

Syncope

DIAGNOSTIC

7.40%	Cardiac Echo
1.60%	Lipid Profile
0.80%	Urinalysis
0.70%	EEG
0.70%	Cardiac Markers
0.70%	Hepatic Function Panel
0.60%	Folate Level
0.60%	Urine Culture
0.40%	Phos
0.30%	Holter Monitor
0.30%	Iron/TIBC
0.10%	Magnesium Level
0.10%	Basic Metabolic
0.00%	Ferritin

RADIOLOGY

1.20%	Carotid Artery Doppler Study
0.90%	Head CT without Contrast
0.20%	Brain MRI with/without Contrast

DRUG

3.00%	Folic Acid
1.80%	Thiamine
1.70%	Fluoxetine
0.60%	Albuterol Nebulizer
0.30%	Ipratropium Nebulizer
0.20%	Ipratropium/Albuterol Inhaler
0.20%	HYDROcodone 5/Acetaminophen 500
0.20%	Aspirin
0.10%	Docusate Na
0.10%	Inhaler Spacer Device
0.10%	Morphine

NURSING

0.70%	Bedside Telemetry
0.10%	Oxygen Therapy

DIET

2.70%	Carbohydrate Counting Diet
0.50%	Regular Diet
0.30%	Restrict Diet for Procedure

TIA/Stroke

DIAGNOSTIC

3.50%	Cardiac Echo
2.40%	Lipid Profile
1.10%	Blood Cell Profile + Plts
0.20%	Diff (peripheral smear)
0.10%	Lytes: Random Urine

GI Bleed

DIAGNOSTIC

8.50%	Blood Cell Profile + Plts
1.60%	Type and Cross
1.10%	Diff (peripheral smear)
0.80%	Phos
0.70%	Protime
0.50%	APTT

RADIOLOGY
2.30% Brain MRI with/without Contrast
1.30% Head MRA without Contrast
1.10% Neck CT Angio with/without Contrast
1.00% Carotid Artery Doppler Study
0.90% Chest Frontal XR
0.90% Head CT Angio with/without Contrast
0.70% Chest PA-Lat XR
0.30% Neck MRA without Contrast

DRUG
1.60% Aspirin
0.80% Intravenous Fluid Orders
0.60% Morphine
0.60% Enoxaparin
0.60% Labetalol
0.50% Esomeprazole
0.30% PrednisoLONE 1% Op
0.30% Furosemide
0.10% NiCARdipine
0.00% Ipratropium Nebulizer

CONSULTS
19.90% Neurology Consult/Appt
1.50% Physical Therapy Consult
1.20% Occupational Therapy Consult
0.60% Speech Pathology
0.00% ACE Team Consult

NURSING
2.20% Bedside Telemetry
0.50% Neurological Checks
0.40% Foley Catheter
0.20% Sequential Compression Device
0.10% TED hose
0.10% Oxygen Therapy
0.00% Net I&O Shift

DIET
30.40% Heart Healthy
0.00% NPO Except Meds

0.40% Magnesium Level
0.10% EKG
0.00% Creatinine: Random Urine

DRUG
1.30% Albuterol Nebulizer
1.10% Ipratropium Nebulizer
1.10% Esomeprazole
0.90% Golytely
0.70% Morphine
0.60% Ondansetron
0.30% Simvastatin
0.30% Octreotide
0.10% Ipratropium/Albuterol Inhaler
0.10% Promethazine

CONSULTS
0.20% Physical Therapy Consult

NURSING
2.60% Hang Blood
0.40% Oxygen Therapy

DIET
3.50% Heart Healthy

Table 10.6. The four admission order set 'drafts' generated by our system for the survey: Congestive Heart Failure, Syncope, TIA/Stroke, and GI Bleed. The sections of each order set is sorted by the probability above the decision threshold. This is shown in the left-hand column of the draft, but this number is not explained to the survey respondents.

3.4. Results. Table 10.7 presents a collation of qualitative responses, which fell into three broad categories comprising six general statements. A summary of how the drafts affected the resulting order set is shown in Table 10.8. This includes, for raters with and without a draft, the average length of their order set, the average time to complete their order set, the average precision and recall of the draft, and the average inter-rater agreement by Fleiss' kappa.

Table 10.9 lists orders which contributed to errors in precision and recall: orders which were included in the draft but universally unwanted and orders not included in the draft wanted by at least two raters. Table 10.10 lists the most helpful orders on the drafts - those that clinicians added at least 20% more frequently when the draft was given.

3.5. Discussion. The order set drafts positively impacted order set generation. Table 10.8 shows the drafts helped participants remember on average 4 items and that they wrote the drafts on average 23 seconds faster. The improvement in precision and recall when the draft was given (average +16% and +25%, respectively) demonstrate the drafts helped participants attentuate to the domain.

Table 10.10 lists orders the drafts helped participants remember. The majority of these are routine but important (IV Fluids, TED hose), but some were surprising mistakes the participants left off (cardiac echo in CHF and EEG in syncope). The reason for this is likely due to the task difficulty (Table 10.7). The majority of these participants were not used to

Statement	Frequency
Drafts are helpful	100%
. These are helpful reminders that help me pick things more quickly.	100%
Drafts are imperfect	100%
. The layout isn't how I think about ordering; I'm used to ADCVANDALISM.	71%
. Some of these are incorrect or are relevant in very specific comorbid situations.	57%
. I can't write an order set without a little more detail on the condition, like upper or lower GI bleed.	43%
Task is difficult	71%
. I don't usually do this by hand; it's strange to not have a computer prompt me.	71%
. I'm not used to doing this without a clinical scenario.	43%

Table 10.7. Summary of qualitative feedback from survey respondents, which fell into three major and six minor categories.

writing generic order sets, even though they were all very experienced with the conditions presented. This likely also explains the relatively low inter-rater agreement (average 0.48).

Although the draft surveys were imperfect (average 48% precision and 64% recall) and cannot be safely used without editing, Table 10.9 demostrates that they would be passable with only a few changes. Many mistakes on the draft are unnecessary but not dangerous (e.g., additional consults), though a few are (e.g., IV fluids in CHF). Many orders not included but used in the final order sets are not critical to diagnosis and treatment (e.g., I/O and heart healthy diet), though again a few are (e.g., aspirin and chest x-ray in CHF). It is interesting that the physicians were not surprised chest x-ray was not included, because it usually is done in the emergency department before admission. Several did not think to include it. This highlights the importance of understanding the environment in which data is being collected.

The worst performance of the drafts was for syncope. Physicians commented both that they know the standard workup for syncope by heart (explaining the lack of speed improvement), and that they hate treating syncope because there are so many possible reasons for it (explaining the low precision). This perhaps highlights that the system works best in situations that are not quite routine but also not so complicated that it becomes impossible to tease apart standard treatments from unusual comorbidities.

Perhaps the most important result is that qualitatively all the raters said they found the drafts helpful (Table 10.7). Many also commented on problems with the drafts, but these were predominantly suggestions to make them more helpful. Many comments on the errors

	CHF		GI Bleed		Stroke		Syncope	
draft given?	N	Y	N	Y	N	Y	N	Y
Length	14.3	18.3	10.3	19.0	18.7	20.5	11.7	12.3
Speed	5:34	4:21	4:38	4:18	4:15	3:50	2:52	3:21
Precision	32.3%	48.4%	20.2%	57.1%	43.1%	53.7%	26.7%	31.7%
Recall	71.8%	82.1%	45.7%	66.3%	80.0%	88.6%	30.5%	22.1%
κ	0.415	0.564	0.538	0.519	0.481	0.432	0.404	0.468

Table 10.8. Comparison of physician choices with our draft order sets. Results are split into the cases where a draft was given and where it was not. For each order set, the average number of orders, speed to write the order set, precision and recall compared to the draft, and interr-rater agreement are shown.

indicated that they did not realize the drafts were automatically generated by a computer from treatment data (e.g., "The extraneous stuff was a little distracting. They're used a lot but only in certain situations. Presenting it all at once makes me feel scatterbrained."). Perhaps this system, if nothing else, has passed the Turing test.

CHF	GI Bleed	Stroke	Syncope
Included but unwanted			
Urine Culture	Creatinine Urine	Morphine	Ferritin
Blood Culture	Albuterol	Prednisolone	Docusate Na
Iron Level	Ipratropium	Ipratropium	Inhaler Spacer
Insulin	Morphine	ACE Consult	Morphine
Albuterol Neb	Simvastatin	Neck CT Angio	Salbutamol
IV Fluids	Salbutamol	Neck MRA	Vicodin
Morphine	Heart Healthy		Holter Monitor
ACE Consult			Iron/TIBC
Renal Consult			Folate Level
OT Consult			Albuterol Neb
PT Consult			Fluoxetine
TED hose			Thiamine
			Low-carb Diet
			Folic Acid
Not included but wanted			
EKG 85.7%	NPO 100.0%	Metabolic Panel 71.4%	Metabolic Panel 85.7%
Aspirin 71.4%	Metabolic Panel 85.7%	Protime 57.1%	EKG 71.4%
Chest X-Ray 57.1%	IV Fluids 71.4%	EKG 57.1%	Heart Healthy 42.9%
Heparin 42.9%	Telemetry 71.4%	Statin 42.9%	
Foley Cath. 28.6%	NG Tube 57.1%	LFTs 28.6%	
Heart Healthy 28.6%	Net IO 42.9%		

Table 10.9. Order set draft items which lowered the precision and recall. Top: orders which were included in the draft but universally unwanted. Bottom: orders not in the draft which were wanted by more than one rater.

CHF	GI Bleed	Stroke	Syncope
Net IO +50.0%	Type and Cross +75.0%	Net IO +66.7%	Lipid Profile +50.0%
O_2 Therapy +50.0%	EKG +75.0%	Enoxaparin +50.0%	Head CT +33.3%
Low Sodium Diet +50.0%	Blood smear +66.7%	Foley Catheter +50.0%	O_2 Therapy +25.0%
Mg Level +41.7%	Mg Level +66.7%	Lipid Profile +33.3%	EEG +25.0%
Hydralazine +33.3%	Promethazine +66.7%	OT Consult +33.3%	
Clopidogrel +33.3%	Octreotide +50.0%	Blood Smear +25.0%	
Glucose Bedside +33.3%	Phos Level +41.7%	Electrolytes +25.0%	
Cardiac Markers +25.0%	O2 Therapy +41.7%	IV Fluids +25.0%	
Cardiac Echo +25.0%	Golytely +33.3%	Esomeprazole +25.0%	
	PT Consult +33.3%	TED hose +25.0%	
	Protime +25.0%	Chest XRay +25.0%	

Table 10.10. Order set draft items which most frequently improved the final order sets. Shown here: orders on the draft included on the final order set at least 20% more frequently when the draft was given. The exact percentage improvement is given next to the order.

CHAPTER 11

Closing Remarks

There is incredible burden in creating and maintaining localized CDS content - so much so that it is frequently not done, at all but a few pioneering institutions. Yet, CDS systems are one of the few aspects of HIT which have demonstrated increased quality of healthcare delivery and reduced costs.

Therefore, in the preceding pages, we have developed an approach to ease the burden of CDS content maintenance, by extracting treatment decisions from local data in the form of situation-specific treatment suggestion lists. Our approach uses Bayesian network learning with several innovative algorithmic advances. Overall, we found that our system can reproduce the local standard of care well. When a chief complaint is known, our system can produce a short menu (on average < 20 items, and in some cases < 5) that will correctly list the most relevant next treatments and tests. When more context is known (previous treatments, comorbid conditions, and demographics) the accuracy of the lists grows and the length shrinks substantially. In such situations, the lists are frequently more than 90% sensitive. Although clinicians were subjectively undecided as to whether the system helped them generate admission order sets, the data show that lists generated with the help of our system were more complete and constructed faster than without.

Domain-specific networks learned with our system produced networks of 80-90 treatments 10-20 comorbid diagnoses, and demographic information in, on average, 50 minutes. This process is automatic and can therefore be scheduled to happen during nights and evenings. Additionally, as long as the structure of the network stays fixed, updating the probabilities in the network is extremely quick (< 2 minutes per network). Once the network is generated, it can be reasoned over (i.e. the context of the hospitalization can be dynamically updated) almost instantaneously. Overall, this approach allows dozens of *partitioned networks*, triggered by chief complaint, to be dynamically loaded to assist with a patient encounter or order set development.

1. Summary of Contributions

We have made the following novel contributions in this work:

- Chapter 3. The idea of using Bayesian structure learning algorithms to generate situation-specific order sets using local data, to promote and simplify the development of CDS. This resulted in complex networks of association able to generate order sets from data with much greater accuracy than better-studied methods.
- Chapter 4. A methodology to generate dynamic situation-specific order sets as the clinical situation evolves, and two statistics that measure the accuracy of such order sets (PPV80 and tAUC). This allows rapid retrospective, data-based analysis of domain-specific networks.
- Chapter 5. A SQL-based approach to rapidly find statistical relationships among the data, which is used in the following two chapters.
- Chapter 6. A method to use these relationships for principled feature selection using network reasoning, which selects more relevant treatments for conditions in domain-specific networks than a frequency-based measure.
- Chapter 7. A method to use the statistical relationships from chapter 5, as well as other advances in Bayesian learning theory, to learn networks with comparable predictive power at much higher speed than previous algorithms, on average by an order of magnitude.
- Chapter 8. A temporal Bayesian network approach for fully-observed systems in which the task is predicting the next event. This approach can be implemented on existing structure learning and inference tools with only a moderate increase in complexity, and applies a concept called 'fading' to learning, which allows learning about past events without unrolling the network (and the associated increased computational complexity).
- Chapters 9 and 10. An exploration of additional data, decision thresholds, and partitioned networks to devise a complete system for situation-specific treatment advice in inpatient medicine.

In addition to these novel contributions, we also thoroughly reviewed the state-of-the-art in clinical data mining and reasoning, theories of crowd wisdom, and Bayesian network learning and reasoning (including existing software tools). We expect that this research will be the beginning of a long line of future improvements.

2. Limitations and Future Directions

2.1. Limitations. This research is predicated on the assumption that average patterns in the data represent reasonably good care for future patients. In many decision-making problems, average patterns do in fact represent 'crowd wisdom,' (Surowiecki, 2005) but 'crowd madness' (the domination of bad decisions in a group) can occur as well. In medicine this is reflected by such phenomena as the widely varying Medicare spending by region (Fisher et al., 2009). Discriminating wisdom from madness is important future work. (See Chapter 1.)

Also, this system relies primarily on order-entry data (previous orders and diagnoses) in the current encounter to drive its treatment advice. A greater patient context is desirable. Many data-sources exist that generate patient data in an encounter. Some of these are frequently recorded (e.g., laboratory results, telemetry results, and active diagnoses from previous hospitalizations), and some are not (e.g., many nursing assessments like vitals and patient condition). Our exploration in Chapter 9 demonstrated that incorporating additional data sources is fraught with complexity, and in earlier work we explored determining which diagnoses remain active across hospitalizations and found this also quite challenging (Klann and Schadow, 2010). Nonetheless this is important future research.

2.2. Future Directions. We feel this work is an early step in utilizing local practice data for crowd-wisdom methodologies. Therefore, the future directions for this work are legion. They can be divided into incremental changes and paradigm shifts. Two non-exhaustive lists follow.

Incremental:

- Develop methods to incorporate longitudinal data into the temporal model.
- Develop new methods to integrate disparate data sources.
- Develop methods to detect synonymous orders by their usage patterns in the system (e.g., various equivalent painkillers).
- Favor relationships in network learning where more consistent data is available.
- Explore a dynamic decision threshold that is not tied to a single, fixed number.
- Choose decision thresholds that assign less importance to common comfort measures.

- Incorporate outcomes so as to filter negative behavior.
- Incorporate information on the physician so as to filter out inexperienced practitioners.
- Investigate applications of other Bayesian-network probability queries (e.g., MAP - maximal a posteriori - queries find a most likely configurations of evidence, which might be useful in suggesting groups of treatments).
- Support multiple orders of the same item within a hospitalization.

Paradigm:

- Implement and evaluate a live decision-support design tool for content maintainers.
- Implement and evaluate a live dynamic-decision support system using our methodology, and dynamically modify it to adapt to the feedback of users.
- Develop quality-monitoring applications using this approach, to detect behavior outside the norm.
- Develop an approach to take snapshots of local standards using this methodology, to compare across regions.

3. Conclusions

We have successfully developed methods to reverse-engineer treatment decisions from local order-entry data, by applying and advancing a technique called Bayesian structure learning. These methods were used to successfully generate accurate situation-specific treatment menus, primarily on inpatient hospitalizations. Moreover, they generated admission order set drafts which aided clinicians in developing a final order set. We feel that methods for using local data to assist the development of decision support and monitoring of practice patterns will become an important area of research in the coming years.

APPENDIX A

Additional Information

1. Why Tetrad and SMILE?

In this work, we provide at best meager evidence that Tetrad and SMILE are the optimal tools for Bayesian network learning and inference. Here we offer a more comprehensive review of available software.

A variety of free and commercial software packages perform Bayesian network structure learning and inference. For the most part, these two tasks are performed by separate applications, and the latter task is better supported. Because this work does not propose new inference methods, we sought an efficient inference tool with an Application Programmer's Interface (API). For structure learning, however, we expected to develop our own algorithms, and so we required either a very fine-grained API (to perform e.g., BDeu scoring) or full source code. Open-source structure-learning tools are quite rare, and such a fine-grained API is unavailable. Additionally, we knew structure learning with edge restrictions would be a useful feature. Also, various tools support a variety of advanced features: causal or temporal inference, inference with uncertain (soft) evidence, varieties of temporal structure learning, etc. Finally, we required tools that could handle large datasets and networks of several hundred nodes.

Kevin Murphy maintains a list of many Bayesian software packages[1], but it is silent on some of these points, so we undertook an evaluation of many current packages. The results of this evaluation can be seen in Table A.1 and Table A.2.

We can immediately eliminate the structure-learning tools that are not open-source, leaving us with four packages. Of these, we must eliminate BNT, because our experiments have shown it cannot handle large networks. (BNT is an excellent package for preliminary experimentation, however, because it touches on nearly every feature that has been researched on Bayesian Networks.) Its architecture relies heavily on objected-oriented Matlab, which cannot be parallelized with the Matlab Distributed Computing Server (MDCS), and one machine cannot handle large networks due to the program's reliance on cell arrays,

[1] http://www.cs.ubc.ca/ murphyk/Bayes/bnsoft.html

which are not efficient in the way Matlab matrices are. Of the remaining three tools, CTBN-RLE only supports continuous time Bayesian networks. In Chapter 8, we elect not to use this model, so this too is out of the running. Remaining are Pebl, bnlearn, and Tetrad. All have slightly different features. Pebl alone supports causal Bayesian networks, another model for handling time. It also provides well documented source code. Unfortunately, it implements the least complex array of learning algorithms. Although it provides a BDeu scorer, it offers no statistical independence tests and only provides a simple best-first greedy search and simplified simulated annealing search. bnlearn provides the best support (among the remaining packages) for hybrid algorithms, implementing a very fast optimized MMHC learner. However, we found Tetrad to be better documented and supported, and it offered the widest variety of building blocks in its source code for developing our own methods. Therefore this was our choice for structure learning, not because it offered everything but because it was the best available.

Selecting an inference tool was simpler. Table A.2 is not an exhaustive list - many fly-by-night tools attempt Bayesian inference. The tools listed here are fairly well supported. We slightly preferred a Java API and a free tool, leaving four options. ACE only supports a particular compiled network approach (decomposing a junction tree into an arithmetic circuit), which did not allow dynamic changes to the network during inference. We did not end up using such functionality, but it seemed unnecessary to limit our options. Logically, Tetrad seemed like a logical choice because we chose to use it for structure learning. However, SMILE was more full-featured, supporting more file formats and inference algorithms. Therefore we chose SMILE. It is worth noting that GEnie is a popular graphical tool for Bayesian inference (in Windows) and it is actually a front-end for SMILE.

2. Table of Synonymous Medications

When cleaning the order data in Chapter 1, we combined orders for medications that only differed in route of administration. Here we list the terms used for this process. These terms were removed from the end of medication names to combine multiple medications

	Netica	BayesiaLab	BNT	LibB	Tetrad	Pebl
Structure	N	Y	Y	F, P	F, (P)	F, P
Scalable	N	(Y)	(Y)	Y	Y	N
Parameters	ML, EM	ML, EM	ML, EM	ML, EM	ML, EM	ML
Uncertain values	Y	N	N	N	N	N
CBN	N	Y	Y	N	N	Y
DBN	N	(Y)	Y	N	N	N
CTBN	N	N	N	N	N	N
Open-source	N	N	Matlab	N	Java	Python
Free	N	N	Y	Y	Y	Y

	SamIAM	CTBN-RLE	Causal Explorer	SMILE	bnlearn
Structure	N	Y	N	F, (P)	F, (P)
Scalable	N	N	Y	N	Y
Parameters	EM	Y	N	ML, EM	ML
Uncertain values	N	N	N	N	N
CBN	N	N	N	N	N
DBN	N	Y	N	N	N
CTBN	N	Y	N	N	N
Open-source	N	C++	N	N	R
Free	Y	Y	Y	Y	Y

Table A.1. Software for Bayesian network learning. P stands for the specification of possible parents, and F stands for the specification of forced or forbidden edges. (P) indicates a limitation in the specification, usually in the form of a limited number of 'temporal tiers.' Note that only Pebl also supports soft structural constraints in the form of energy matrices. Scalable indicates the type of algorithms implemented: N indicates only score-based, (Y) indicates constraint-based, and Y indicates hybrid. In parameter learning, ML is the Maximum Likelihood algorithm and EM is expectation maximization, used on incomplete data. CBN, DBN, and CTBN are methods for handling time (Causal Bayesian Network, Dynamic Bayesian Network, and Continuous Time Bayesian Network, respectively).

into single items. These terms were chosen manually by skimming a list of all medications used in Gopher and can be seen in Table A.3.

	Netica	BayesiaLab	BNT	Tetrad	SMILE
Exact	Y	Y	Y	Y	Y
Approximate	Y	Y	Y	Y	Y
Compiled	Y	N	N	N	Y
Soft evidence	Y	Y	Y	N	Y
API	Java	N	Matlab	Java	Java/C/.NET
Free	N	N	Y	Y	Y

	SamIAM	CTBN-RLE	ACE	BUGS
Exact	Y	Y	N	N
Approximate	N	Y	N	Y
Compiled	N	N	Y+	N
Soft evidence	N	N	Y	(N)
API	N	C++	Java	Java, R
Free	Y	Y	Y	Y

Table A.2. Software packages for Bayesian network inference.

Search terms used to eliminate redundant medications						
'iv'	'suppository'	'suspension'	'solution'			
'inj'	'sr'	'concentrate'	'susp'	'syrup'	'powder'	
'liquid'	'drops'	'drip'	'enema'	'patch'	'ointment'	
'infusion'	'cap'	'tab'	'xr'	'chewable'	'ir'	'elixir'
'cream'	'pwd'	'syr'	'topical'	'lotion'	'lot'	'tabs'
'drip'	'oint'	'gel'	'liquid'	'sl'	'chew'	'nasal'
'patch'	'immun'	'vaccine'	'inhaler'	'shampoo'		
'irrigation'	'chewable'	'suppository'				
'buffered'	'(enteric-coated)'					

Table A.3. Medication words used in Gopher to indicate a route of administration.

BIBLIOGRAPHY

Abidi, S. S. R. and S. Manickam (2002, December). Leveraging XML-based electronic medical records to extract experiential clinical knowledge: An automated approach to generate cases for medical case-based reasoning systems. International Journal of Medical Informatics 68(1-3), 187–203.

Agrawal, R., R. Srikant, J. B. Bocca, M. Jarke, and C. Zaniolo (1994). Fast algorithms for mining association rules. In Proceedings of the 20th Int. Conf. Very Large Data Bases,VLDB, pp. 487–499. Morgan Kaufmann.

Aliferis, C., I. Tsamardinos, and A. Statnikov (2003). HITON: a novel markov blanket algorithm for optimal variable selection. In Proceedings of the AMIA Symposium, Volume 2003, pp. 21–25.

Althoff, K., R. Bergmann, S. Wess, M. Manago, E. Auriol, O. I. Larichev, A. Bolotov, Y. I. Zhuravlev, and S. I. Gurov (1998, January). Case-based reasoning for medical decision support tasks: The inreca approach. Artificial Intelligence in Medicine 12(1), 25–41.

Arrow, K. J. (1950). A difficulty in the concept of social welfare. The Journal of Political Economy 58(4), 328–346.

Ash, J. S., P. N. Gorman, V. Seshadri, and W. R. Hersh (2004). Computerized physician order entry in U.S. hospitals: results of a 2002 survey. J Am Med Inform Assoc 11(2), 95–9.

Austen-Smith, D. and J. S. Banks (1996, March). Information aggregation, rationality, and the Condorcet jury theorem. The American Political Science Review 90(1), 34–45.

Barnett, G. O., J. J. Cimino, J. A. Hupp, and E. P. Hoffer (1987, July). DXplain. an evolving diagnostic decision-support system. JAMA: The Journal of the American Medical Association 258(1), 67–74.

Beinlich, I., H. J. Suermondt, R. M. Chavez, and G. Cooper (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In Proceedings of the Second European Conference on Artificial Intelligence in Medicine, Volume 256.

Bellazzi, R. and B. Zupan (2008, February). Predictive data mining in clinical medicine: Current issues and guidelines. International Journal of Medical Informatics 77(2), 81–97.

Berner, E. (2009, June). Clinical Decision Support Systems: State of the Art. AHRQ Publication No. 09-0069-EF.

Bichindaritz, I. and C. Marling (2006, February). Case-based reasoning in the health sciences: What's next? Artificial Intelligence in Medicine 36(2), 127–135.

Blumenthal, D. and M. Tavenner (2010, August). The meaningful use regulation for electronic health records. New England Journal of Medicine 363(6), 501–504.

Boxwala, A. A., M. Peleg, S. Tu, O. Ogunyemi, Q. T. Zeng, D. Wang, V. L. Patel, R. A. Greenes, and E. H. Shortliffe (2004, June). GLIF3: a representation format for sharable computer-interpretable clinical practice guidelines. Journal of Biomedical Informatics 37(3), 147–161.

Boyen, X. and D. Koller (1998). Tractable inference for complex stochastic processes. In Proceedings of the UAI, Volume 98.

Buhmann, M. D., P. Melville, V. Sindhwani, N. Quadrianto, W. L. Buntine, L. Torgo, X. Zhang, P. Stone, J. Struyf, H. Blockeel, K. Driessens, R. Miikkulainen, E. Wiewiora, J. Peters, R. Tedrake, N. Roy, J. Morimoto, P. A. Flach, and J. Frnkranz (2011). ROC analysis. In C. Sammut and G. I. Webb (Eds.), Encyclopedia of Machine Learning, pp. 869–875. Boston, MA: Springer US.

Buntine, W. (1991). Theory refinement on bayesian networks. In Proceedings of the seventh conference (1991) on Uncertainty in artificial intelligence UAI, Los Angeles, California, United States, pp. 52–60. Morgan Kaufmann Publishers Inc.

Cao, H., G. B. Melton, M. Markatou, and G. Hripcsak (2008, December). Use abstracted patient-specific features to assist an information-theoretic measurement to assess similarity between medical cases. Journal of Biomedical Informatics 41(6), 882–888.

Carter, J. S., S. H. Brown, M. S. Erlbaum, W. Gregg, P. L. Elkin, T. Speroff, and M. S. Tuttle (2002). Initializing the VA medication reference terminology using UMLS metathesaurus co-occurrences. Proceedings of the AMIA Annual Symposium., 116–120.

Chavira, M., A. Darwiche, and M. Jaeger (2006, May). Compiling relational bayesian networks for exact inference. International Journal of Approximate Reasoning 42(1-2), 4–20.

Che, C. (2007, May). Mining frequent order sets. Master's thesis, University of Utah.

Chen, E. S., G. Hripcsak, H. Xu, M. Markatou, and C. Friedman (2008, January). Automated acquisition of Disease-Drug knowledge from biomedical and clinical documents: An initial study. Journal of the American Medical Informatics Association 15(1), 87–98.

Chickering, D. M. (2003). Optimal structure identification with greedy search. J. Mach. Learn. Res. 3, 507–554.

Chickering, D. M., D. Heckerman, and C. Meek (2004). Large-sample learning of bayesian networks is NP-hard. The Journal of Machine Learning Research 5, 1287–1330.

Conati, C., A. S. Gertner, K. VanLehn, and M. J. Druzdzel (1997). On-line student modeling for coached problem solving using bayesian networks. In Proceedings of the Sixth International Conference on User Modeling (UM–96), Vienna, New York, pp. 231–242. Springer Verlag.

Condorcet, M. (1785). Essay sur l'application de l'analyse de la probabilit des decisions: Redues et pluralit des voix. l'Imprimerie Royale.

Conn, J. (2007, October). Failure, de-installation of EHRs abound: study. Modern Healthcare Online.

Cooper, G. and C. Yoo (1999). Causal discovery from a mixture of experimental and observational data. In Proceedings of the Fifthteenth Conference on Uncertainty in Artificial Intelligence (UAI99), pp. 116–125.

Cooper, G. F. (1990). Probabilistic inference using belief networks is NP-hard. Artificial Intelligence 42(2-3), 393–405.

de Dombal, F. T., D. J. Leaper, J. R. Staniland, A. P. McCann, and J. C. Horrocks (1972, April). Computer-aided diagnosis of acute abdominal pain. British Medical Journal 2(5804), 9–13.

Dean, T. and K. Kanazawa (1989). A model for reasoning about persistence and causation. Computational Intelligence 5(2), 142–150.

Druzdzel, M. J. (1999). SMILE: structural modeling, inference, and learning engine and GeNIe: a development environment for graphical decision-theoretic models. In Proceedings of the 16th national conference on Artificial intelligence and the 11th Innovative applications of artificial intelligence conference, AAAI '99/IAAI '99, Menlo Park, CA, USA, pp. 902–903. American Association for Artificial Intelligence.

Druzdzel, M. J. and H. J. Suermondt (1994). Relevance in probabilistic models:Backyards in a small world. In Working notes of the AAAI-1994 Fall Symposium Series: Relevance, Volume 101.

Eaton, D. and K. Murphy (2007). Exact bayesian structure learning from uncertain interventions. In AI & Statistics, Volume 2, pp. 107–114.

Elidan, G. and S. Gould (2008). Learning bounded treewidth bayesian networks. Journal of Machine Learning Research 9, 2699–2731.

Ely, J. W., W. Levinson, N. C. Elder, A. G. Mainous, and D. C. Vinson (1995, April). Perceived causes of family physicians' errors. The Journal of Family Practice 40(4), 337–344.

Fienberg, S. E. (2007, July). The Analysis of Cross-Classified Categorical Data. Springer.

Fisher, E., D. Goodman, J. Skinner, and K. Bronner (2009, February). Health care spending, quality, and outcomes.

Ford, E. W., N. Menachemi, and M. T. Phillips (2006). Predicting the adoption of electronic health records by physicians: when will health care be paperless? J Am Med Inform Assoc 13, 106–112.

Friedman, N., K. Murphy, and S. Russell (1998). Learning the structure of dynamic probabilistic networks. In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98), pp. 139–147.

Friedman, N., I. Nachman, and D. Peer (1999). Learning bayesian network structure from massive datasets: The sparse candidate algorithm. In Proceedings of the UAI.

Frost, J., S. Okun, T. Vaughan, J. Heywood, and P. Wicks (2011, January). Patient-reported outcomes as a source of evidence in Off-Label prescribing: Analysis of data from PatientsLikeMe. Journal of Medical Internet Research 13(1).

Garg, A. X., N. K. J. Adhikari, H. McDonald, M. P. Rosas-Arellano, P. J. Devereaux, J. Beyene, J. Sam, and R. B. Haynes (2005, March). Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: A systematic review. JAMA 293(10), 1223–1238.

Geissbuhler, A. and R. A. Miller (1999). Distributing knowledge maintenance for clinical decision-support systems: the "knowledge library" model. In Proceedings of the AMIA symposium, pp. 770. American Medical Informatics Association.

Glover, F. (1990). Tabu search, part II. ORSA Journal on computing 2(1), 4–32.

Goldberg, D., D. Nichols, B. M. Oki, and D. Terry (1992). Using collaborative filtering to weave an information tapestry. Commun. ACM 35(12), 61–70.

Hanley, J. A. and B. J. McNeil (1982, April). The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143(1), 29–36.

Haug, J. D. (1997, July). Physicians' preferences for information sources: a meta-analytic study. Bulletin of the Medical Library Association 85(3), 223–232.

Hayes, P. J. and J. McCarthy (1969). Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligence 4, 463–502.

Heckerman, D. (1998). A tutorial on learning with bayesian networks. Learning in graphical models (89), 301.

Heckerman, D. E. and B. N. Nathwani (1992). Toward normative expert systems: Part II. probability-based representations for efficient knowledge acquisition and inference. Methods of Information in medicine 31(2), 106–116.

Hedberg, S. (1998, March). Is AI going mainstream as last? a look inside microsoft research. IEEE Intelligent Systems 13(2), 21–25.

Himmelstein, D. U., A. Wright, and S. Woolhandler (2010, January). Hospital computing and the costs and quality of care: A national study. The American Journal of Medicine 123(1), 40–46.

Hoot, N. and D. Aronsky (2005). Using bayesian networks to predict survival of liver transplant patients. AMIA Annual Symposium Proceedings, 345–349.

Horvitz, E. (1998). Lumiere project: Bayesian reasoning for automated assistance. Decision Theory & Adaptive Systems Group, Microsoft Research. Microsoft Corp. Redmond, WA.

Hripcsak, G., P. Ludemann, T. A. Pryor, O. B. Wigertz, and P. D. Clayton (1994). Rationale for the arden syntax. Computers and Biomedical Research 27(4), 324.

Institute of Medicine (1991). The computer-based patient record: an essential technology for health care. Washington, DC: National Academy Press.

Jensen, F. V. and T. D. Nielsen (2007). Bayesian networks and decision graphs. Springer.

Kaushal, R., A. K. Jha, C. Franz, J. Glaser, K. D. Shetty, T. Jaggi, B. Middleton, G. J. Kuperman, R. Khorasani, M. Tanasijevic, and D. W. Bates (2006). Return on investment for a computerized physician order entry system. J Am Med Inform Assoc 13(3), 261–6.

Kaushal, R., K. G. Shojania, and D. W. Bates (2003). Effects of computerized physician order entry and clinical decision support systems on medication safety: a systematic review. Arch Intern Med 163(12), 1409–1416.

King, A. J. and G. Cowlishaw (2007, April). When to use social information: the advantage of large group size in individual decision making. Biology Letters 3(2), 137–139.

Klann, J., G. Schadow, and J. M. McCoy (2009). A recommendation algorithm for automating corollary order generation. In Proceedings of the AMIA Symposium, pp. 333–337.

Klann, J. G. and G. Schadow (2010). Modeling the information-value decay of medical problems for problem list maintenance. In Proceedings of the 1st ACM International Health Informatics Symposium, pp. 371–375.

Kohavi, R. and G. H. John (1997, December). Wrappers for feature subset selection. Artificial Intelligence 97(1-2), 273–324.

Koller, D. and N. Friedman (2009, November). Probabilistic graphical models: principles and techniques. MIT Press.

Koller, D. and M. Sahami (1996). Toward optimal feature selection. In Machine Learning – International Workshop then Conference, pp. 284–292.

Kopach-Konrad, R., M. Lawley, M. Criswell, I. Hasan, S. Chakraborty, J. Pekny, and B. N. Doebbeling (2007, November). Applying systems engineering principles in improving health care delivery. Journal of General Internal Medicine 22(S3), 431–437.

Lallich, S., O. Teytaud, and E. Prudhomme (2007). Association rule interestingness: measure and statistical validation. Quality Measures in Data Mining, 251.

Larntz, K. (1978, June). Small-Sample comparisons of exact levels for chi-squared goodness-of-fit statistics. Journal of the American Statistical Association 73(362), 253–263.

Liang, S., S. Fuhrman, and R. Somogyi (1998). Reveal, a general reverse engineering algorithm for inference of genetic network architectures. Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing, 18–29.

Lindberg, D. A. and B. L. Humphreys (1995). The high-performance computing and communications program, the national information infrastructure and health care. Journal of the American Medical Informatics Association 2(3), 156.

Linden, G., B. Smith, and J. York (2003). Amazon. com recommendations: Item-to-Item collaborative filtering. IEEE Internet Computing, 76–80.

Lucena, B. (2003). A new lower bound for tree-width using maximum cardinality search. SIAM Journal on Discrete Mathematics 16(3), 345–353.

MacDonald, C. J., W. M. Tierney, J. M. Overhage, D. K. Martin, B. Smith, C. Wodniak, L. Blevins, J. Warvel, J. Warvel, and J. Meeks-Johnson (1994). The regenstrief medical record system–experience with MD order entry and community-wide extensions. Proceedings of the Annual Symposium on Computer Application in Medical Care, 1059–1059.

Maciejewski, R., S. Rudolph, R. Hafen, A. Abusalah, M. Yakout, M. Ouzzani, W. S. Cleveland, S. J. Grannis, M. Wade, and D. S. Ebert (2008). Understanding syndromic hotspots - a visual analytics approach. IEEE Symposium on Visual Analytics Science and Technology.

Mani, S., C. Aliferis, S. Krishnaswami, and T. Kotchen (2007). Learning causal and predictive clinical practice guidelines from data. Studies in Health Technology and Informatics 129(Pt 2), 850–854.

Margaritis, D. and S. Thrun (1999). Bayesian network induction via local neighborhoods. In Advances in Neural Information Processing Systems 12.

McDonald, C. (2002). The regenstrief medical record system 2002: Focus on the medical gopher clinical workstation. In Proceedings of the AMIA Symposium, pp. 1216.

McDonald, C. J. (1976, December). Protocol-based computer reminders, the quality of care and the non-perfectability of man. The New England Journal of Medicine 295(24), 1351–5.

McDonald, C. J., J. M. Overhage, M. Barnes, G. Schadow, L. Blevins, P. R. Dexter, and B. Mamlin (2005). The indiana network for patient care: a working local health information infrastructure. Health Affairs 24(5), 1214.

McGlynn, E. A., S. M. Asch, J. Adams, J. Keesey, J. Hicks, A. DeCristofaro, and E. A. Kerr (2003, June). The quality of health care delivered to adults in the united states. N Engl J Med 348(26), 2635–2645.

Medeiros, A. K., B. F. Dongen, and W. M. Aalst (2004). Process mining: extending the alpha-algorithm to mine short loops. In BETA Working Paper Series, EIndhoven University of Technology.

Medicare & Medicaid Services, C. f. (2006). National Health Expenditure Projections 2007-2017. 7500 Security Boulevard Baltimore, MD 21244: Office of the Actuary: Centers for Medicare & Medicaid Services.

Melton, G. B., S. Parsons, F. P. Morrison, A. S. Rothschild, M. Markatou, and G. Hripcsak (2006, December). Inter-patient distance metrics using SNOMED CT defining relationships. Journal of Biomedical Informatics 39(6), 697–705.

Middleton, B. (2009). Clinical decision support consortium. http://www.partners.org/cird/cdsc/.

Miller, R., F. E. Masarie, and J. D. Myers (1986, October). Quick medical reference (QMR) for diagnostic assistance. M.D. Computing: Computers in Medical Practice 3(5), 34–48.

Murphy, K. et al. (2001). The bayes net toolbox for matlab. Computing science and statistics 33(2), 1024–1034.

Murphy, K. P. (2002). Dynamic bayesian networks: representation, inference and learning. Ph. D. thesis, University of California, Berkeley.

Nilsson, R., J. M. Pe na, J. Bjorkegren, and J. Tegnor (2007). Consistent feature selection for pattern recognition in polynomial time. The Journal of Machine Learning Research 8, 589–612.

Nodelman, U., D. Koller, and C. R. Shelton (2005). Expectation propagation for continuous time bayesian networks. In UAI, Volume 5, pp. 431–440.

Nodelman, U., C. R. Shelton, and D. Koller (2002). Continuous time bayesian networks. In Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence, pp. 378–387.

Nodelman, U., C. R. Shelton, and D. Koller (2003). Learning continuous time bayesian networks. In Proceedings of the Nineteenth International Conference on Uncertainty in Artificial Intelligence, Volume 451458.

Orszag, P. (2008, January). Growth of Health Care Costs. Second and D Streets, S.W.; Washington, D.C. 20515.

Overhage, J. M., W. M. Tierney, X. A. Zhou, and C. J. McDonald (1997, October). A randomized trial of corollary orders to prevent errors of omission. Journal of the American Medical Informatics Association 4(5), 364–375.

OMadadhain, J., D. Fisher, P. Smyth, S. White, and Y. B. Boey (2005). Analysis and visualization of network data using JUNG. Journal of Statistical Software 10, 1–35.

Page, L., S. Brin, R. Motwani, and T. Winograd (1999, November). The PageRank citation ranking: Bringing order to the web. http://ilpubs.stanford.edu:8090/422/.

Paterson, G., S. S. R. Abidi, and S. D. Soroka (2005). HealthInfoCDA: case composition using electronic health record data sources. Studies in Health Technology and Informatics 116, 137–142.

Payne, T. H. (2000). Computer decision support systems. Chest 118(2 suppl), 47S–52S.

Pearl, J. (1988). Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann.

Pearl, J. (2000). Causality: models, reasoning, and inference. Cambridge University Press.

Pellet, J. P. and A. Elisseeff (2008). Using markov blankets for causal structure learning. The Journal of Machine Learning Research 9, 1295–1342.

Perley, C. M. (2006, April). Physician use of the curbside consultation to address information needs: report on a collective case study. Journal of the Medical Library Association 94(2), 137–144.

Perrier, E., S. Imoto, and S. Miyano (2008). Finding optimal bayesian network given a super-structure. Journal of Machine Learning Research 9(10), 2251–2286.

Popescu, M. and G. Arthur (2006). OntoQuest: a physician decision support system based on ontological queries of the hospital database. AMIA Annual Symposium Proceedings, 639–643.

Pradhan, M., M. Henrion, G. Provan, B. Del Favero, and K. Huang (1996, August). The sensitivity of belief networks to imprecise probabilities: an experimental investigation. Artificial Intelligence 85(1-2), 363–397.

Quinlan, J. R. (1993). C4. 5: programs for machine learning. Morgan Kaufmann.

Ramoni, R. B., N. L. Saccone, D. K. Hatsukami, L. J. Bierut, and M. F. Ramoni (2009, January). A testable prognostic model of nicotine dependence. Journal of Neurogenetics 23(3), 283–292.

Ramsey, J. (2011). Tetrad project homepage. http://www.phil.cmu.edu/projects/tetrad/tetrad4.html.

Sanders, D. L. and D. Aronsky (2006). Prospective evaluation of a bayesian network for detecting asthma exacerbations in a pediatric emergency department. In Proceedings of the AMIA Symposium, Volume 2006, pp. 1085–1085.

Santangelo, J., P. Rogers, J. Buskirk, H. S. Mekhjian, J. Liu, and J. Kamal (2007). Using data mining tools to discover novel clinical laboratory test batteries. AMIA Annual Symposium Proceedings, 1101.

Schafer, J. B., J. A. Konstan, and J. Riedl (2001, January). E-Commerce recommendation applications. Data Mining and Knowledge Discovery 5(1), 115–153.

Shelton, C. R., Y. Fan, W. Lam, J. Lee, and J. Xu (2010). Continuous time bayesian network reasoning and learning engine. The Journal of Machine Learning Research 11, 1137–1140.

Shortliffe, E. H. (1976). Computer-based medical consultations, MYCIN. Elsevier.

Shwe, M. A., B. Middleton, D. E. Heckerman, M. Henrion, E. J. Horvitz, H. P. Lehmann, and G. F. Cooper (1991). Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. i. the probabilistic model and inference algorithms. Methods of information in Medicine 30(4), 241.

Sidl, C. and A. Lukcs (2006). Shaping sql-based frequent pattern mining algorithms. Knowledge Discovery in Inductive Databases, 188–201.

Simon, S. R., R. Kaushal, P. D. Cleary, C. A. Jenter, L. A. Volk, E. J. Orav, E. Burdick, and D. W. Bates (2007). Physicians and electronic health records. Arch Intern Med 167, 507–512.

Simon, S. R., C. S. Soran, R. Kaushal, C. A. Jenter, L. A. Volk, E. Burdick, P. D. Cleary, E. Orav, E. G. Poon, and D. W. Bates (2009, July). Physicians' use of key functions in electronic health records from 2005 to 2007: A statewide survey. Journal of the American Medical Informatics Association 16(4), 465–470.

Sittig, D. F., A. Wright, J. A. Osheroff, B. Middleton, J. M. Teich, J. S. Ash, E. Campbell, and D. W. Bates (2008, April). Grand challenges in clinical decision support. Journal of Biomedical Informatics 41(2), 387–392.

Spirtes, P. and C. Glymour (1991). An algorithm for fast recovery of sparse causal graphs. Social Science Computer Review 9(1), 62.

Spirtes, P., C. N. Glymour, and R. Scheines (2000). Causation, prediction, and search. MIT Press.

Surowiecki, J. (2005, August). The wisdom of crowds. Random House, Inc.

Tawfik, A. Y. (1997). Changing times: an investigation in probabilistic temporal reasoning. Ph. D. thesis, University of Saskatchewan.

Tawfik, A. Y. and E. Neufeld (1994). Temporal bayesian networks. In Proceedings of the TIME-94–International Workshop on Temporal Representation and Reasoning, pp. 85–92.

Toussi, M., J. Lamy, P. Le Toumelin, and A. Venot (2009). Using data mining techniques to explore physicians' therapeutic decisions when clinical guidelines do not provide recommendations: methods and example for type 2 diabetes. BMC Medical Informatics and Decision Making 9(1), 28.

Tsamardinos, I. and C. F. Aliferis (2003). Towards principled feature selection: Relevancy, filters and wrappers. In Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics.

Tsamardinos, I., C. F. Aliferis, and A. Statnikov (2003). Algorithms for large scale markov blanket discovery. In The 16th International FLAIRS Conference, Volume 103.

Tsamardinos, I., L. E. Brown, and C. F. Aliferis (2006, March). The max-min hill-climbing bayesian network structure learning algorithm. Machine Learning 65(1), 31–78.

van der Sijs, H., J. Aarts, A. Vulto, and M. Berg (2006, March). Overriding of drug safety alerts in computerized physician order entry. Journal of the American Medical Informatics Association 13(2), 138–147.

van Dijk, T., J. P. van den Heuvel, and W. Slob (2006). Computing treewidth with LibTW.

Vojtech, H. and J. Starren (2009). EHR data pre-processing facilitating process mining: an application to chronic kidney disease. AMIA Annual Symposium Proceedings 2009, 889.

Vomlel, J. (2006). Noisy-or classifier. International Journal of Intelligent Systems 21(3), 381–398.

Waitman, L. R. (2004, June). Pragmatics of implementing guidelines on the front lines. Journal of the American Medical Informatics Association 11(5), 436–438.

Wang, D. Z., M. J. Franklin, M. Garofalakis, and J. M. Hellerstein (2010). Querying probabilistic information extraction. Proceedings of the VLDB Endowment 3(1).

Wasserman, L. (2004, September). All of statistics: a concise course in statistical inference. Springer.

Wendykier, P. and J. G. Nagy (2010, September). Parallel colt: A High-Performance java library for scientific computing and image processing. ACM Transactions on Mathematical Software (TOMS) 37, 31:1–31:22.

White, S. and P. Smyth (2003). Algorithms for estimating relative importance in networks. In Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data mining, pp. 266–275.

Wright, A., E. S. Chen, and F. L. Maloney (2010, December). An automated technique for identifying associations between medications, laboratory results and problems. Journal of Biomedical Informatics 43(6), 891–901.

Wright, A. and D. F. Sittig (2006). Automated development of order sets and corollary orders by data mining in an ambulatory computerized physician order entry system. AMIA Annual Symposium Proceedings, 819–23.

Zaki, M. (2000). Scalable algorithms for association mining. Knowledge and Data Engineering, IEEE Transactions on 12(3), 372–390.

Zhou, L., C. S. Soran, C. A. Jenter, L. A. Volk, E. J. Orav, D. W. Bates, and S. R. Simon (2009, August). The relationship between electronic health record use and quality of care over time. Journal of the American Medical Informatics Association: JAMIA 16(4), 457–464.

CURRICULUM VITAE

Jeffrey G. Klann

EDUCATION:

PhD in Health Informatics, Indiana University (IUPUI), Indianapolis, IN (2007-2011)

- PhD, Awarded September 2011

BS and MEng in Computer Science and Engineering, MIT (Massachusetts Institute of Technology), Cambridge, MA (1997-2004)

- Master of Engineering, Awarded June 2004; Major: Computer Science; GPA: 4.9 [5.0 scale]
- Bachelor of Science, Awarded May 2001; Major: Computer Science; GPA: 4.5 [5.0 scale]
- Minor: Theater Arts
- Master's Thesis Title: "Run Manager Module for CORAL Laboratory Management"
- Undergraduate Thesis Title: "Data Mastermind of the Information Ball"
- Thesis Advisors: Vicky Diadiuk (for MEng), Rick Borovoy (for BS)

HONORS AND AWARDS:

- AMIA Student Paper Competition: Second Place 2010, Finalist 2009
- JAMIA Student Editorial Board Member 2009-2010
- Regenstrief NLM Training Program 2008-2011 (First pre-doctoral student ever selected. Program included stipend and additional education.)
- Indiana University Student Fellow Award 2007, School of Informatics Scholarship 2008-2011 (Included tuition and stipend.)

PROFESSIONAL ORGANIZATIONS:

- American Medical Informatics Association
- Association of Computing Machinery
- Sigma Xi Scientific Research Society

RESEARCH INTERESTS:

- AI and data mining for dynamic, contextualized information in healthcare applications.
- Developing methods to utilize the growing mass of healthcare data for physician cognitive supports.
- Using data to sort, prioritize, and customize the user experience in healthcare applications.
- Injecting web interface motifs (e.g., ubiquitous search) into healthcare applications.

- Real-time AI and natural language processing to simplify healthcare workflow.
- The personal health record as collaborative social media.

RESEARCH EXPERIENCE:

Regenstrief Institute: Graduate Research Fellow in Medical Informatics (Fall 2007-Summer 2011)

MIT Clinical Decision Making Group: Research Assistant (Fall 2006-Summer 2007)

MIT Microsystems Technology Laboratory (MTL): Research Assistant (2002)

MIT Media Laboratory: Epistemology and Learning Group Research Assistant (Fall 1999-Spring 2001)

TEACHING EXPERIENCE:

IUPUI, Invited Lecture for Medical Informatics Seminar (April 2011)

IUPUI, Instructor for Database Module in Medical Informatics Elective (Aug. 2009-Sept. 2011)

MIT, Teaching Assistant for Introduction to Computer Science (Fall Term 2001)

MIT, Tutor for Introduction to Computer Science (Fall Term 1998)

MAJOR SERVICE ACTIVITIES:

JAMIA Student Editorial Board (2009-2010)

Ad-hoc Reviewer: Journal of Biomedical Informatics, AMIA Annual Symposium (2009-2011)

Graduate Vice President of the Informatics School Government at IUPUI (2008-2011)

Cooperative Community Organizer in Boston (2005-2007)

Intervarsity Christian Fellowship at MIT: Bible Study Leader (1999-2001)

Alpha Delta Phi Fraternity at MIT: Several leadership roles (1998-2001)

SCHOLARSHIP:

Peer-reviewed Papers:

Klann, Schadow, Downs. "A Method to Compute Treatment Suggestions from Local Order Entry Data," Nov 2010, AMIA Fall Symposium 2010.

- Student Paper Award: Second Place

Klann, Schadow. "Modeling the Information-Value Decay of Medical Problems for Problem List Maintenance," Nov 2010, ACM International Health Informatics Symposium 2010.

Klann, Schadow, McCoy. "A Recommendation Algorithm for Automating Corollary Order Generation," Nov 2009, AMIA Fall Symposium 2009.

- Student Paper Award Finalist

Klann, Szolovits. "An Intelligent Listening Framework for Capturing Encounter Notes from a Doctor-Patient Dialog," Nov 2009, BMC Medical Informatics and Decision Making, Vol.9, Supp. 1.

Klann, Szolovits. "An Intelligent Listening Framework for Capturing Encounter Notes from a Doctor-Patient Dialog," November, 2008, IEEE Conference of Bioinformatics and Biomedicine (Biomedical and Health Informatics Workshop). [Later revised and published as the 2009 BMC paper above]

Borovoy, Silverman, Gorton, Klann, Notowidigdo, Knep, Resnick. "Folk Computing: Revisiting Oral Tradition as a Scaffold for Co-Present Communities," April, 2001, Proceedings of the ACM CHI 2001 Human Factors in Computing Systems Conference.

Abstracts and Posters:

Klann. "Applications of Network Analysis to Clinical Data," October, 2011, AMIA Fall Symposium 2011. (in press)

Klann, Chang, McCoy. "Building Search Functionality in an Electronic Medical Record System," November, 2010, AMIA Fall Symposium 2010.

Klann, McCoy. "Using Structured Clinical Documents to Provide Enhanced Display of Patient Records," November, 2008, AMIA Fall Symposium 2008. [Live demo online.]

Klann, McCoy. "Learning from the Crowd: Integrating local standards of care into a CDSS system for medication orders," May, 2008, AMIA Spring Congress 2008.

Klann, McCoy. "Back to the future: revisiting the command line in conceptualizing EHR interfaces," December, 2007, NSF Biomedical Informatics Workshop.

Invited Presentations:

"A System to Translate Physician Actions into Decision Support," June 2011, National Library Of Medicine Student Conference

"Learning from the Crowd: On the Marquis De Condorcet and the Reverend Bayes," March 2011, Regenstrief Institute Work In Progress

"Curbside to Bedside: Extracting Decision Support from Medical Order-Entry Data," June 2010, Regenstrief Institute Work In Progress

"Clinical Recommendation Algorithms for Corollary Ordering," June 2009, National Library Of Medicine Student Conference

"From Web To Bedside: Exploring Applications of Internet Solutions to Healthcare," June 2009, Regenstrief Institute Work In Progress

PROFESSIONAL EXPERIENCE:

nTAG Interactive, Boston, MA (2003-2005, part-time)

Trilogy, Automotive Development Division, Austin, TX (Fall 2002 and Summer 2001)

Microsoft, Consumer Windows Division, Seattle, WA (Summer 2000)

Lucent Technologies, Kenan Division (Summer 1999)

MIT, Computer Help Desk (Spring 1999, part-time)

American Chemical Society, Chemical Abstracts, Columbus OH (Summer 1998)

Computer Instruction, Cleveland and Columbus OH (1994-1997, part-time)

TECHNICAL SKILLS:

- Software development: Team and large project design, analysis, data mining and Bayesian networks, adapting legacy code, Web applications and services, client/server programming, GUIs, OOP.
- Programming languages (tiered by experience level):
  - Tier 1: SQL, Java, Python
  - Tier 2: HTML, PHP, C, C++, Groovy
  - Tier 3: Javascript, JSP, BASIC, AWK, ANT, Matlab, Unix shell scripts
- IT: Hardware admin and software maintenance. (Unix/Linux, Win7-95, Mac OSX/OS9)
- Video Editing / Desktop Publishing: Final Cut Pro, Avid Pinnacle Studio, Quark, Pagemaker
- Coursework: Java design, data structures, algorithms, A.I., computer systems engineering, compiler design, graphics, expert systems, distributed systems, computability/complexity, biomedical informatics, health standards and terminologies, social aspects of information systems, clinical information systems, probability/statistics/discrete math, research design, complex systems, pedagogy, data mining, Bayesian networks

ARTS AND PERFORMANCE:

- Education: MIT theater department undergraduate minor (1997-2001), ImprovBoston University training program (2004)
- Directing: One full-length production and three student-written one-act plays at MIT (1998, 2004, 2005)
- Teaching: Instructor/curriculum designer for MIT high-school studies program course on improvisation (1998, 2001)
- Performance: Improvisational comedy performer at MIT (Roadkill Buffet '97-'02) and ImprovBoston (The Postmodern Avengers '04-'07), Major roles in six plays at MIT ('98-'06)
- Film: Editing, acting, and writing for short-film competitions (The Postmodern Avengers '04-'06)