Università degli studi di Modena e Reggio Emilia
Dipartimento di Ingegneria “Enzo Ferrari”

Tesi di Laurea Magistrale in Ingegneria Informatica

Bayesian Networks: Optimization of the Human-Computer Interaction process in a Big Data Scenario

Candidato:
Emanuele Charalambis

Relatore:
Sonia Bergamaschi
University of Modena and Reggio Emilia

Correlatore:

H. V. Jagadish
University of Michigan

Anno Accademico 2012 – 2013

ABSTRACT

The process of interaction between man and computer, better known as Human-Computer Interaction (HCI) is a crucial part in the design of every information system. The quality of the system depends on its usability or how it is represented and used by users. Therefore, the design and optimization of HCI and User Experience (UX) are topics of great interest in the field of research. The growth in HCI field has not only been in quality of interaction, it has also experienced different branching in its history. Instead of designing regular interfaces, the different research branches have had different focus on the concepts of multimodality rather than unimodality, intelligent adaptive interfaces rather than command/action based ones, and finally active rather than passive interfaces. The ubiquitous computing is trying to incorporate the technology into the environment in order to make it natural and invisible at the same time. This thesis has the task of explaining the main topics that were addressed during my research period in the department of computer science at the University of Michigan: Analysis of theory and practical implementation of Probabilistic Graphical Models (PGMs) with special attention to Bayesian Networks in order to develop intelligent algorithms for the optimization of the user experience in a Big Data visualization scenario. A key part of the research project was the analysis and solution of the case study of Faceted Browsing. In the Big Data era the dynamic context entails many problems in the areas of research and data visualization, therefore is evident the need to develop more effective and efficient solutions that are able to handle this complexity. Having to deal with an increasingly wide and varied amount of data while providing solutions that favor the simplicity of the user experience lead to the need for synergy between the research in the field of computer science and in the field of statistics. In this thesis I focused my attention in the study of the probabilistic graphical models (PGMs), in particular of Bayesian networks, to provide a new paradigm of information systems design which optimizes the process of interaction between human and computer (HCI). The case study that was analyzed was the Faceted Browsing accessing technique: ~~In~~ in a Big Data scenario the standard visualization systems are unable to efficiently manage the query search and visualization of an increasing number of data that speedly vary, at the expense of the user experience that can be slow and complex. Therefore, it is necessary to provide adaptive and intelligent interfaces that guide the user through every step of its query search and visualization of the resultset.

In this thesis, I will show how I used the knowledge gained on the PGMs to implement a software optimization that makes the user experience, from the search to the display of data, as easy and efficient as possible. I have designed and developed a solution that optimizes the Faceted Browsing search in ***Apache Solr***~~,~~ (a widely used enterprise search platform), by creating an algorithm that, based on the features that describe a dataset, indicates the most significant description categories of ~~of~~ the data, and a query suggestion algorithm that guides the user's query.

OUTLINE OF THE THESIS:

The Thesis is organized as follows:

- **Chapter 1:** Overview on the Human-Computer Interaction (HCI) and User Experience (UX).
- **Chapter 2:** Overview on Big Data, the consequences of the advent of a similar scenario in Data Visualization, and an analysis of existing technologies.

- **Chapter 3:** Introduction on probabilistic graphical models, their use and their functionality.
- **Chapter 4:** Overview on Bayesian networks, their structure and topology, their use and the most used inference algorithms.
- **Chapter 5:** Overview of existing softwares that permit to manage and model Bayesian networks.
- **Chapter 6:** Detailed description of ***OpenMarkov***, the software I used to implement a solution to the case study I dealt with, its API and the automatic learning of Bayesian networks from a dataset.
- **Chapter 7:** Description of the case study, namely the Faceted Browsing, and how to optimize the UX using the Bayesian networks, describing from the theory to the software implementation of the solution and part of the algorithms I used.
- **Conclusions:** Conclusions concerning the thesis and description of any ideas for future work.

This thesis is part of a joint project between UNIMORE (University of Modena and Reggio Emlia) and UMich (University of Michigan), in which two research thesis were conducted in parallel allowing us to work in synergy in the actual implementation of the solution for the proposed case study. The two thesis are intended to join the study and analysis of the Apache Solr search platform and the study and analysis of Bayesian networks to optimize the interaction between the user and the information system of Faceted Browsing. In this project, the role of this thesis is the study of Bayesian networks in a Big Data enviroment and how they can be used to improve the search and display functionalities ~~and~~ of data in the information systems of today. The second thesis research, conducted by Paolo Malavolta, is instead based on the study and analysis of Faceted Browsing in a Big Data scenario using Apache Solr platform. The joint work has allowed us to implement a solution for the Apache Solr's front-end.

SOMMARIO

Il processo di interazione tra uomo e computer, conosciuto meglio come Human-Computer Interaction (HCI) è una parte fondamentale nella progettazione di ogni sistema. La qualità del sistema dipende dalla sua usabilità ovvero da come viene rappresentato e utilizzato dagli utenti. Pertanto, la progettazione e l'ottimizzazione dell'HCI e della User Experience (UX) sono temi di forte interesse nel campo della ricerca. L'evoluzione nel campo dell'HCI non riguarda solamente la qualità di interazione ma particolare interesse è rivolto alla progettazione di interfacce adattative intelligenti, invece che passive e statiche, che rendano l'esperienza dell'utente più semplice ed efficiente possibile. L'ubiquitous computing sta cercando di incorporare la tecnologia nell'ambiente in modo da renderla naturale e invisibile allo stesso tempo. Questa tesi ha il compito di descrive i principali temi che ho affrontato e analizzato durante il mio periodo di ricerca presso il dipartimento di computer science alla University of Michigan: Dallo studio e analisi delle teoria dei modelli grafici probabilistici (PGMs) alla effettiva implementazione software di una soluzione basata sull'utilizzo delle reti Bayesiane al fine di sviluppare algoritmi intelligenti per l'ottimizzazione della UX in uno scenario di visualizzazione Big Data. Una parte fondamentale del progetto di ricerca in cui è stato messo in pratica quanto appreso sui modelli statistici è stata l'analisi e soluzione del caso di studio del Faceted Browsing. Nell'era dei Big Data il contesto fortemente dinamico comporta numerosi problemi in ambito di ricerca e visualizzazione dei dati, devono quindi essere fornite soluzioni sempre più efficaci ed efficienti che siano in grado di gestire questa complessità. Il dover trattare una mole sempre più ampia e variegata di dati e fornire al contempo delle soluzioni che favoriscano la semplicità dell'esperienza utente comportano la necessità di una sinergia tra la ricerca in ambito computer science e in ambito statistico-probabilistico. In questa tesi di ricerca ho affrontato lo studio dei modelli grafici probabilistici (PGMs), in particolare delle reti Bayesiane, per fornire un nuovo paradigma di progettazione di interfacce di visualizzazione per sistemi informativi che ottimizzi il processo di interazione tra uomo e computer (HCI).

Il caso di studio che è stato affrontato è stato il Faceted Browsing: In uno scenario Big Data i sistemi di visualizzazione standard risultano incapaci di gestire in maniera efficiente la ricerca e la visualizzazione di un numero sempre maggiore di dati che variano con rapidità, a scapito dell'esperienza utente che può risultare lenta e complessa. E' necessario quindi fornire delle interfacce adattative e intelligenti che guidino l'utente in ogni passo della sua ricerca e visualizzazione del resultset. In questa tesi mostrerò come ho utilizzato le conoscenze acquisite sui PGMs per implementare una ottimizzazione software che renda l'esperienza utente, dalla ricerca alla visualizzazione dei dati, il più semplice ed efficace possibile. Ho sviluppato una soluzione che ottimizza il ***Faceted Browsing*** in ***Apache Solr***, piattaforma di ricerca enterprise largamente utilizzata, realizzando un algoritmo che, in base alle features che descrivono un dataset, indichi le categorie più significative di descrizione dei dati, e un algoritmo di suggerimento nella query di ricerca dell'utente.

DESCRIZIONE SOMMARIA DELLA TESI

La tesi è organizzata come segue:

- **Capitolo 1:** Panoramica generale sulla Human-Computer Interaction (HCI) e sulla User Experience (UX).

- **Capitolo 2:** Panoramica sui Big Data, cosa comporta l'avvento di uno scenario simile nella Data Visualization e le tecnologie esistenti di analisi.
- **Capitolo 3:** Introduzione sui modelli grafici probabilistici, il loro uso e la loro funzionalità.
- **Capitolo 4:** Panorama sulle reti Bayesiane, la loro struttura e topologia, il loro uso e gli algoritmi di inferenza più utilizzati.
- **Capitolo 5:** Panoramica sui software esistenti che permettono di gestire e modellare reti Bayesiane.
- **Capitolo 6:** Descrizione dettagliata di ***OpenMarkov***, il software che ho utilizzato per implementare una soluzione al caso di studio affrontato, la sua API e il learning automatico di reti Bayesiane partendo da un dataset qualsiasi.
- **Capitolo 7:** Descrizione del caso di studio affrontato, ovvero il Faceted Browsing, e come ottimizzare la UX utilizzando le reti Bayesiane, dalla teoria all'implementazione software descrivendo anche parte degli algoritmi utilizzati.
- **Conclusioni:** Conclusioni riguardanti la tesi e descrizione di qualche idea per il lavoro futuro.

Questa tesi è parte di un progetto congiunto tra UNIMORE (Università degli Studi di Modena e Reggio Emlia) e UMich (University of Michigan), in cui due tesi di ricerca sono state condotte in parallelo permettendoci di lavorare in sinergia nell'effettiva implementazione della soluzione al caso di studio proposto. Le due tesi hanno l'obiettivo di congiungere lo studio e analisi della piattaforma di ricerca Apache Solr e lo studio e analisi delle reti Bayesiane per ottimizzare l'interazione tra utente e il sistema informativo di Faceted Browsing. In questo progetto il ruolo svolto da questa tesi è quello dello sullo studio delle reti Bayesiane in ambienti Big Data e di come possano essere utilizzate per migliorare le funzionalità di ricerca e visualizzazione dati nei sistemi informativi odierni. La seconda tesi di ricerca, realizzata da Paolo Malavolta, è invece basata sullo studio e analisi del Faceted Browsing in un ambiente Big Data utilizzando la piattaforma di Apache Solr. Il lavoro congiunto ci ha permesso di implementare un miglioramento del front-end di Apache Solr.

ACKNOWLEDGEMENTS

First and foremost I would like to thank my thesis coordinator, Sonia Bergamaschi, for her unwavering support throughout the duration of my time as a graduate student at University of Modena and Reggio Emilia. His cheerfulness and encouragement have been essential in giving me the space that allowed me to discover my academic interests and passions. I would also like to thank my thesis advisor H. V. Jagadish for the great experience and the support he offered me during my internship at University of Michigan. I kindly thank Datariver for the partial economic support to organize the internship.

I owe special thanks to my family, for their useful and constructive comments and advice in every choice I made in my life.

COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY of MICHIGAN

October 29, 2013

To whom it may concern:

I am pleased to extend an invitation to Mr. Emanuele Charalambis, a Master student at the University of Modena e Reggio Emilia, Modena, Italy to visit the University of Michigan at Ann Arbor, Department of Computer Science and Engineering, during the period of November 1, 2013 to January 29, 2014 to conduct collaborative research in the field of Big Data management, faceted browsing and Bayesian networks.

The travelling expenses will be covered by a fund provided by Professor Sonia Bergamaschi of Department of Engineering "Enzo Ferrari", University of Modena e Reggio Emilia, Modena, Italy.

During his visit, Mr. Charalambis will be self-financed.

I will not provide financial support but I will provide research facilities in my lab to conduct collaborative research.

Sincerely,

H V Jagadish
Bernard A Galler Collegiate Professor of
Elec. Engg. and Computer Science.
University of Michigan
2260 Hayward Ave
Ann Arbor, MI 48109-2121

COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY of MICHIGAN

February 25, 2014

To whom it may concern:

This is to confirm that Mr. Emanuele Charalambis spent three months from November 2013 through January 2014 working on research under my direction in the Software Systems Research Laboratory located in the Bob and Betty Beyster Building at the University of Michigan.

During this period, he worked on research problems in Big Data management, faceted browsing and Bayesian networks, he participated in research meetings of the DataBase group, and he attended various seminars that were organized in the DataBase group or in Computer Science and Engineering department.

Sincerely,

H V Jagadish
Bernard A Galler Collegiate Professor of
Elec. Engg. and Computer Science.
University of Michigan
2260 Hayward Ave
Ann Arbor, MI 48109-2121

GLOSSARY

HCI	Human–Computer Interaction (HCI) involves the study, planning, design and uses of the interaction between people (users) and computers. It is often regarded as the intersection of computer science, behavioral sciences, design and several other fields of study. The term connotes that, unlike other tools with only limited uses a computer has many affordances for use and this takes place in an open-ended dialog between the user and the computer.
FACETS	Facets correspond to properties of the information elements. They are often derived by analysis of the text of an item using entity extraction techniques or from pre-existing fields in a database such as author, descriptor, language, and format. Thus, existing web-pages, product descriptions or online collections of articles can be augmented with navigational facets.
FACETED BROWSING	Faceted search, also called faceted navigation or faceted browsing, is a technique for accessing information organized according to a faceted classification system, allowing users to explore a collection of information by applying multiple filters. A faceted classification system classifies each information element along multiple explicit dimensions, called facets, enabling the classifications to be accessed and ordered in multiple ways rather than in a single, pre-determined, taxonomic order.
BIG DATA	Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. The challenges include capture, storage, search, sharing, transfer, analysis and visualization. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to spot business trends, determine quality of research, prevent diseases, link legal citations, combat crime, and determine real-time roadway traffic conditions.
VISUALIZATION	Visualization (of Big Data) refers to the implementation of more contemporary visualization techniques to illustrate the relationships within data. Visualization tactics include applications that can display real-time changes and more illustrative graphics, thus going beyond pie, bar and other charts. These illustrations veer away from the use of hundreds of rows, columns and attributes toward a more artistic visual representation of the data.
PGMs	Probabilistic Graphical Models (PGMs) are probabilistic models for which a graph denotes the conditional dependence structure between random variables. They are commonly used in probability theory, statistics—particularly Bayesian statistics—and machine learning. PGMs use a graph-based representation as the foundation for encoding a complete distribution over a multi-dimensional space and a graph that is a compact or factorized representation of a set of independences that hold in the specific distribution. Two branches of graphical representations of distributions are commonly used, namely, Bayesian networks and Markov networks. Both families encompass the properties of factorization and independences, but they differ in the set of independences they can encode and the factorization of the distribution that they induce.

CONDITIONAL PROBABLITY	In probability theory, a conditional probability measures the probability of an event given that (by assumption, presumption, assertion or evidence) another event has occurred. If the events are A and B respectively, this is said to be "the probability of A given B". It is commonly denoted by P(A|B), or sometimes PB(A). The concept of conditional probability is one of the most fundamental and one of the most important concepts in probability theory.
INFERENCE	Inference is the act or process of deriving logical conclusions from premises known or assumed to be true. The conclusion drawn is also called an idiomatic. The laws of valid inference are studied in the field of logic. Alternatively, inference may be defined as the non-logical, but rational means, through observation of patterns of facts, to indirectly see new meanings and contexts for understanding. Of particular use to this application of inference are anomalies and symbols. Inference, in this sense, does not draw conclusions but opens new paths for inquiry. In this definition of inference, there are two types of inference: inductive inference and deductive inference.
BAYESIAN NETS	A Bayesian network is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Formally, Bayesian networks are DAGs whose nodes represent random variables in the Bayesian sense: they may be observable quantities, latent variables, unknown parameters or hypotheses. Edges represent conditional dependencies; nodes that are not connected represent variables that are conditionally independent of each other. Each node is associated with a probability function that takes as input a particular set of values for the node's parent variables and gives the probability of the variable represented by the node.
OPENMARKOV	OpenMarkov is a software tool for probabilistic graphical models (PGMs) developed by the Research Centre on Intelligent Decision-Support Systems of the UNED in Madrid, Spain. It has been designed for: editing and evaluating several types of several types of PGMs, such as Bayesian networks, influence diagrams, factored Markov models, etc.; learning Bayesian networks from data interactively; cost-effectiveness analysis.
SOLR	Solr (pronounced "solar") is an open source enterprise search platform from the Apache Lucene project. Its major features include full-text search, hit highlighting, faceted search, dynamic clustering, database integration, and rich document (e.g., Word, PDF) handling. Providing distributed search and index replication, Solr is highly scalable. Solr is written in Java and runs as a standalone full-text search server within a servlet container such as Apache Tomcat or Jetty. Solr uses the Lucene Java search library at its core for full-text indexing and search, and has REST-like HTTP/XML and JSON APIs that make it usable from most popular programming languages. Solr's powerful external conFiguretion allows it to be tailored to many types of application without Java coding, and it has a plugin architecture to support more advanced customization.
IDE	An integrated development environment (IDE) or interactive development environment is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs offer Intelligent code completion features.

NETBEANS	NetBeans is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5. It is also an application platform framework for Java desktop applications and others. The NetBeans IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans Platform (including the NetBeans IDE itself) can be extended by third party developers.
NOTEPAD++	Notepad++ is a text editor and source code editor for Windows. It aims to be a lightweight and robust editor for a variety of programming and scripting languages. One advantage of Notepad++ over the built-in Windows text editor Notepad, is that Notepad++ supports tabbed editing, which allows working with multiple open files in a single window. Notepad++ opens large files significantly faster and can be used as a replacement for Windows Notepad.
TOMCAT	Apache Tomcat (or simply Tomcat, formerly also Jakarta Tomcat) is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run in. Apache Tomcat includes tools for conFiguretion and management, but can also be configured by editing XML conFiguretion files.
SERVLET CONTAINER	The Servlet container is the component of a web server that interacts with Java servlets. A web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.
AJAX	Ajax (also AJAX; /ˈeɪdʒæks/; an acronym for Asynchronous JavaScript and XML) is a group of interrelated web development techniques used on the client-side to create asynchronous web applications. With Ajax, web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page. Data can be retrieved using the XMLHttpRequest object. Despite the name, the use of XML is not required (JSON is often used instead. See AJAJ), and the requests do not need to be asynchronous.
JSON	JSON or JavaScript Object Notation, is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML. Although originally derived from the JavaScript scripting language, JSON is a language-independent data format, and code for parsing and generating JSON data is readily available in a large variety of programming languages.
REST	Representational state transfer (REST) is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements. REST has been applied to describe desired web architecture, to identify existing problems, to compare alternative solutions, and to ensure that protocol extensions would not violate the core constraints that make the Web successful. The REST architectural style is also applied to the development of Web services as an alternative to other distributed-computing specifications such as SOAP

DBMS	A database is an organized collection of data. The data are typically organized to model relevant aspects of reality in a way that supports processes requiring this information. For example, modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies. Database management systems (DBMSs) are specially designed software applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is a software system designed to allow the definition, creation, querying, update, and administration of databases.
METADATA	Metadata is "data about data". The term is ambiguous, as it is used for two fundamentally different concepts (types). Structural metadata is about the design and specification of data structures and is more properly called "data about the containers of data"; descriptive metadata, on the other hand, is about individual instances of application data, the data content.
GUI	In computing, graphical user interface GUI is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLI), which require commands to be typed on the keyboard.
XML	Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all free open standards.

CONTENTS

LIST OF FIGURES

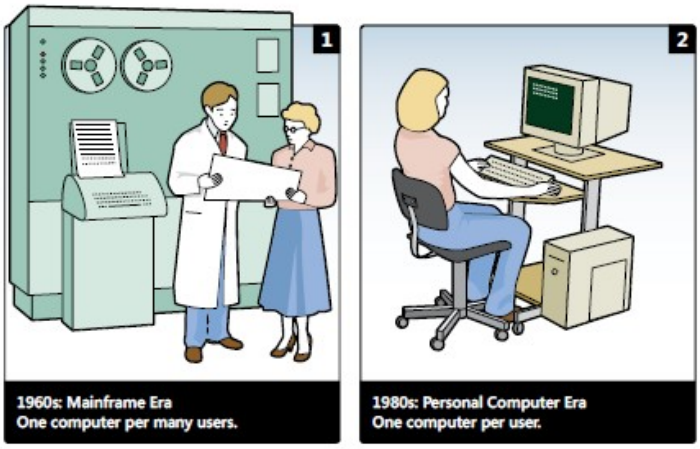
1 HUMAN-COMPUTER INTERACTION

Utilizing computers had always begged the question of interfacing. The methods by which human has been interacting with computers has travelled a long way. The journey still continues and new designs of technologies and systems appear more and more every day and the research in this area has been growing very fast in the last few decades. The growth in Human-Computer Interaction (HCI) field has not only been in quality of interaction, it has also experienced different branching in its history [CITATION Emm06 \l 1040]. Instead of designing regular interfaces, the different research branches have had different focus on the concepts of multimodality rather than unimodality, intelligent adaptive interfaces rather than command/action based ones, and finally active rather than passive interfaces. This paper intends to provide an overview on the state of the art of HCI and in the next section, basic definitions and terminology of HCI are given. Then an overview of existing technologies and also generic recent advances in the field is provided. This is followed up by a description on the different architectures of HCI designs. The final sections pertain to description on some of the applications of HCI and future directions in the field.

1.1 Our changing world

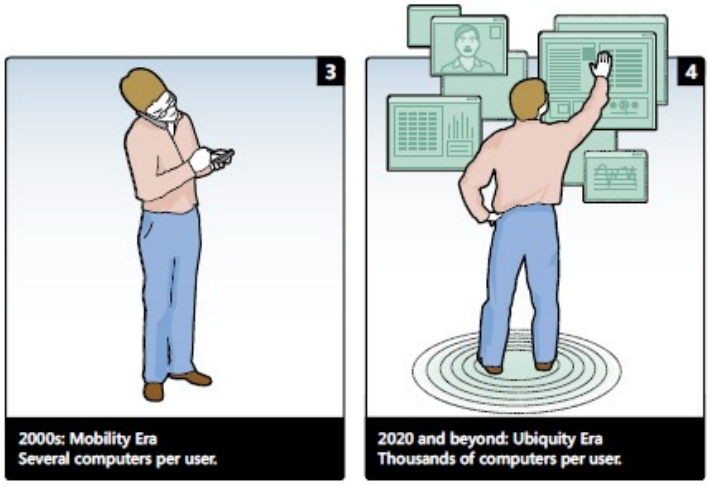
Major changes have occurred within the computer revolution; changes which encompass all aspects of its role. These are not just quantitative in nature, such as exponential increases in processing power and storage capacity, but are more fundamental, pointing not only to the function of computer technology, but its emerging diversity both in terms of its form and place in the world. Computers are now embedded within a huge range of materials and artefacts, and take on roles in almost all aspects of life. People and lifestyles are altering.These changes are sometimes spurred on by technology, but other times work in parallel or provoke technological innovation. There is a global scale of change which is taking place hand in hand with new technologies. This gives rise to tensions between individuals and governments, and between globalisation and cultural diversity. In this Part, we comment on change at all levels, and provide pointers to where we are going in future.

1.2 Changing computers



There have been various computer-driven revolutions in the past: the widespread introduction of the personal computer (PC) was one, the invention of the graphical browser was another, and the Internet yet another. There have also been computer eras where one type of computer has dominated, having straightforward implications for whether the computers were shared or personal, and whether they were specialised for commodities or not (see diagram below). But the ways computers have altered our lives, all aspects of our lives, is more comprehensive than, at first blush, recollections of these technological revolutions or eras might suggest. Computers affect how we undertake the most prosaic of activities – from buying food to paying our bills – and they do so in ways we might not have imagined when the first personal computers arrived on our desks. They have also created wholly new experiences, for example, allowing us to inhabit virtual worlds with people from many different parts of the globe. In between these extremes, from the prosaic to the wholly new, computers have taken over from older technologies in ways that looked merely like substitution at first but which have ended up creating radical change.

Photography, for example, has retained its familiarity despite moving from being chemically-based to being digital. At the point of creation, people still 'point and shoot' in much the same way as they used to. However, what one can do with images when they are digital is quite different.



Whereas, before, we may have only printed one or two rolls of film, displaying the photos on the mantelpiece or in an album, digital images are now reproduced many times over, and are often broadcast around the world on websites The activities we undertake and the goals we have in mind when we take photos and share them, then, are not at all the same now as they were even five years ago. It is not just in terms of user experiences, such as shopping, games, and picture-taking that the world has changed. Computers have altered our sense of the world at large, letting us see images of far-away places, instantaneously and ubiquitously. The world, now, seems so much smaller than it was even a decade ago. In this section we begin to look at many different aspects of how computing technologies havechanged and their impact on our lives.

1.3 Human-Computer Interaction: Definition, Terminolgy

Sometimes called as Man-Machine Interaction or Interfacing, concept of Human-Computer Interaction/Interfacing (HCI) was automatically represented with the emerging of computer, or more generally machine, itself. The reason, in fact, is clear: most sophisticated machines are worthless unless they can be used properly by men. This basic argument simply presents the main terms that should be considered in the design of HCI: *functionality* and *usability [CITATION DTe07 \l 1040]*. Why a system is actually designed can ultimately be defined by what the system can do i.e. how the functions of a system can help towards the achievement of the purpose of the system. *Functionality* of a system is defined by the set of actions or services that it provides to its users. However, the value of functionality is visible only when it becomes possible to be efficiently utilised by the user. *Usability* of a system with a certain functionality is the range and degree by which the system can be used efficiently and adequately to accomplish certain goals for certain users. The actual effectiveness of a system is achieved when there is a proper balance between the functionality and usability. Having these concepts in mind and considering that the terms computer, machine and system are often used interchangeably in this context, HCI is a design that should produce a fit between the user, the machine and the required services in order to achieve a certain performance both in quality and optimality of the services. Determining what makes a certain HCI design good is mostly subjective and context dependant. For example, an aircraft part designing tool should provide high precisions in view and design of the parts while a graphics editing software may not need such a precision. The available technology could also affect how different types of HCI are designed for the same purpose. One example is using commands, menus, graphical user interfaces (GUI), or virtual reality to access functionalities of any given computer. In the next section, a more detailed overview of existing methods and devices used to interact with computers and the recent advances in the field is presented.

1.4 Overview on Human-Computer Interaction

The advances made in last decade in HCI have almost made it impossible to realize which concept is fiction and which is and can be real. The thrust in research and the constant twists in marketing cause the new technology to become available to everyone in no time. However, not all existing technologies are accessible and/or affordable by public. In the first part of this section, an overview of the technology that more or less is available to and used by public is presented. In the second part, an outlook of the direction to which HCI research is heading has been drawn.

1.5 Intelligent and Adaptive HCI

Although the devices used by majority of public are still some kind of plain command/action setups using not very sophisticated physical apparatus, the flow of research is directed to design of intelligent and adaptive interfaces. The exact theoretical definition of the concept of intelligence or being smart is not known or at least not publicly agreeable. However, one can define these concepts by the apparent growth and improvement in functionality and usability of new devices in market.

As mentioned before, it is economically and technologically crucial to make HCI designs that provide easier, more pleasurable and satisfying experience for the users. To realize this goal, the interfaces are getting more natural to use every day. Evolution of interfaces in note-taking tools is a good example. First there were typewriters, then keyboards and now touch screen tablet PCs that you can write on using your own handwriting and they recognize it change it to text [CITATION GRi05 \l 1040] and if not already made, tools that transcript whatever you say automatically so you do not need to write at all. One important factor in new generation of interfaces is to differentiate between using intelligence in the making of the interface (Intelligent HCI) [CITATION MTM98 \l 1040] or in the way that the interface interacts with users (Adaptive HCI) [CITATION AKi06 \l 1040]. Intelligent HCI designs are interfaces that incorporate at least some kind of intelligence in perception from and/or response to users. A few examples are speech enabled interfaces that use natural language to interact with user and devices that visually track user's movements or gaze and respond accordingly.

Adaptive HCI designs, on the other hand, may not use intelligence in the creation of interface but use it in the way they continue to interact with users. An adaptive HCI might be a website using regular GUI for selling various products. This website would be adaptive -to some extent- if it has the ability to recognize the user and keeps a memory of his searches and purchases and intelligently search, find, and suggest products on sale that it thinks user might need. Most of these kinds of adaptation are the ones that deal with cognitive and affective levels of user activity. Another example that uses both intelligent and adaptive interface is a PDA or a tablet PC that has the handwriting recognition ability and it can adapt to the handwriting of the logged in user so to improve its performance by remembering the corrections that the user made to the recognised text.

Finally, another factor to be considered about intelligent interfaces is that most non-intelligent HCI design are passive in nature i.e. they only respond whenever invoked by user while ultimate intelligent and adaptive interfaces tend to be active interfaces. The example is smart billboards or advertisements that present themselves according to users' taste. In the next section, combination of different methods of HCI and how it could help towards making intelligent adaptive natural interfaces is discussed.

1.6 Ubiquitous Computing and Ambient Intelligence

The latest research in HCI field is unmistakably *ubiquitous computing* (Ubicomp) [CITATION Gre06 \l 1040]. The term which often used interchangeably by *ambient intelligence* and *pervasive computing*, refers to the ultimate methods of human-computer interaction that is the deletion of a desktop and embedding of the

computer in the environment so that it becomes invisible to humans while surrounding them everywhere hence the term ambient. The idea of ubiquitous computing was first introduced by Mark Weiser during his tenure as chief technologist at Computer Science Lab in Xerox PARC in 1998. His idea was to embed computers everywhere in the environment and everyday objects so that people could interact with many computers at the same time while they are invisible to them and wirelessly communicating with each other [CITATION GRi05 \l 1040]. Ubicomp has also been named the Third Wave of computing.

The First Wave was the mainframe era, many people one computer. Then it was the Second Wave, one person one computer which was called PC era and now Ubicomp introduces many computers one person era [CITATION GRi05 \l 1040]. Figure 1-1 shows the major trends in computing

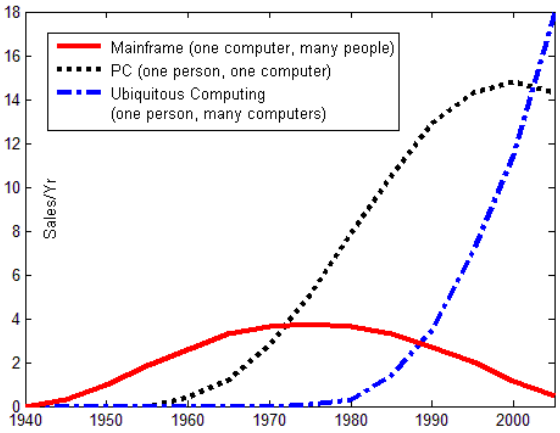


Figure 1-1 Major trends in computing

1.7 HCI Systems Architecture

Most important factor of a HCI design is its conFiguretion. In fact, any given interface is generally defined by the number and diversity of inputs and outputs it provides. Architecture of a HCI system shows what these inputs and outputs are and how they work together. Following sections explain different conFiguretions and designs upon which an interface is based.

1.8 Unimodal HCI Systems

As mentioned earlier, an interface mainly relies on number and diversity of its inputs and outputs which are communication channels that enable users to interact with computer via this interface. Each of the different independent single channels is called a modality [CITATION AJa07 \l 1040]. A system that is based on only one modality is called unimodal. Based on the nature of different modalities, they can be divided into three categories:

1 – Visual-Based

2 – Audio-Based

3 – Sensor-Based

The next sub-sections describe each category and provide examples and references to each modality.

1.8.1 Visual-Based HCI

The visual based human computer interaction is probably the most widespread area in HCI research. Considering the extent of applications and variety of open problems and approaches, researchers tried to tackle different aspects of human responses which can be recognized as a visual signal. Some of the main research areas in this section are as follow:

• Facial Expression Analysis
• Body Movement Tracking (Large-scale)
• Gesture Recognition
• Gaze Detection (Eyes Movement Tracking)

While the goal of each area differs due to applications, a general conception of each area can be concluded. Facial expression analysis generally deals with recognition of emotions visually. Body movement tracking and gesture recognition are usually the main focus of this area and can have different purposes but they are mostly used for direct interaction of human and computer in a command and action scenario. Gaze detection is mostly an indirect form of interaction between user and machine which is mostly used for better understanding of user's attention, intent or focus in context-sensitive situations. The exception is eye tracking systems for helping disabilities in which eye tracking plays a main role in command and action scenario, e.g. pointer movement, blinking for clicking. It is notable that some researchers tried to assist or even replace other types of interactions (audio-, sensor-based) with visual approaches. For example, lip reading or lip movement tracking is known to be used as an influential aid for speech recognition error correction.

1.8.2 Audio-Based HCI

The audio based interaction between a computer and a human is another important area of HCI systems. This area deals with information acquired by different audio signals. While the nature of audio signals may not be as variable as visual signals but the information gathered from audio signals can be more trustable, helpful, and is some cases unique providers of information. Research areas in this section can be divided to the following parts:

- Speech Recognition
- Speaker Recognition
- Auditory Emotion Analysis
- Human-Made Noise/Sign Detections (Gasp, Sigh, Laugh, Cry, etc.)
- Musical Interaction

Historically, *speech recognition* and *speaker recognition* have been the main focus of researchers. Recent endeavors to integrate human emotions in intelligent human computer interaction initiated the efforts in analysis of emotions in audio signals. Other than the tone and pitch of speech data, typical human auditory signs such as sigh, gasp, and etc helped emotion analysis for designing more intelligent HCI system. Music generation and interaction is a very new area in HCI with applications in art industry which is studied in both audio- and visual-based HCI systems.

1.8.3 Sensor-Based HCI

This section is a combination of variety of areas with a wide range of applications. The commonality of these different areas is that at least one physical sensor is used between user and machine to provide the interaction. These sensors as shown below can be very primitive or very sophisticated:

1. Pen-Based Interaction
2. Mouse & Keyboard
3. Joysticks
4. Motion Tracking Sensors and Digitizers
5. Haptic Sensors
6. Pressure Sensors
7. Taste/Smell Sensors

Some of these sensors have been around for a while and some of them are very new technologies. Pen-Based sensors are specifically of interest in mobile devices and are related to pen gesture and handwriting recognition areas. Motion tracking sensors/digitizers are state-of-the-art technology which revolutionized movie, animation, art, and video-game industry. They come in the form of wearable cloth or joint sensors and made computers

much more able to interact with reality and human able to create their world virtually. Haptic and pressure sensors are of special interest for applications in robotics and virtual reality. New humanoid robots include hundreds of haptic sensors that make the robots sensitive and aware to touch. These types of sensors are also used in medical surgery application. A few research works are also done on area of taste and smell sensors; however they are not as popular as other areas.

1.9 Multimodal HCI Systems

The term multimodal refers to combination of multiple modalities. In MMHCI systems, these modalities mostly refer to the ways that the system responds to the inputs, i.e. communication channels [CITATION AJa07 \l 1040]. The definition of these channels is inherited from human types of communication which are basically his senses: Sight, Hearing, Touch, Smell, and Taste. The possibilities for interaction with a machine include but are not limited to these types [CITATION Jen02 \l 1040]. Therefore, a multimodal interface acts as a facilitator of human-computer interaction via two or more modes of input that go beyond the traditional keyboard and mouse. The exact number of supported input modes, their types and the way in which they work together may vary widely from one multimodal system to another. Multimodal interfaces incorporate different combinations of speech, gesture, gaze, facial expressions and other non-conventional modes of input. One of the most commonly supported combinations of input methods is that of gesture and speech.

Although an ideal multimodal HCI system should contain a combination of single modalities that interact correlatively, the practical boundaries and open problems in each modality oppose limitations on the fusion of different modalities. In spite of all progress made in MMHCI, in most of existing multimodal systems, the modalities are still treated separately and only at the end, results of different modalities are combined together. The reason is that the open problems in each area are yet to be perfected meaning that there is still work to be done to acquire a reliable tool for each sub-area. Moreover, roles of different modalities and their share in interplay are not scientifically known. “Yet, people convey multimodal communicative signals in a complementary and redundant manner. Therefore, in order to accomplish a human-like multimodal analysis of multiple input signals acquired by different sensors, the signals cannot be considered mutually independently and cannot be combined in a context-free manner at the end of the intended analysis but, on the contrary, the input data should be processed in a joint feature space and according to a context-dependent model. In practice, however, besides the problems of context sensing and developing context-dependent models for combining multisensory information, one should cope with the size of the required joint feature space. Problems include large dimensionality, differing feature formats, and time-alignment.”

An interesting aspect of multimodality is the collaboration of different modalities to assist the recognitions. For example, lip movement tracking (visual-based) can help speech recognition methods (audio-based) and speech recognition methods (audio-based) can assist command acquisition in gesture recognition (visual-based). The next section shows some of application of intelligent multimodal systems.

1.9.1 Applications

A classic example of a multimodal system is the "*Put That There*" demonstration system. This system allowed one to move an object into a new location on a map on the screen by saying "put that there" while pointing to the object itself then pointing to the desired destination. Multimodal interfaces have been used in a number of applications including map-based simulations, such as the aforementioned system; information kiosks, such as AT&T's *MATCHKiosk* [CITATION MJo04 \l 1040] and biometric authentication systems.

Multimodal interfaces can offer a number of advantages over traditional interfaces. For one thing, they can offer a more natural and user-friendly experience. For instance, in a real-estate system called Real Hunter, one can point with a finger to a house of interest and speak to make queries about that particular house. Using a pointing gesture to select an object and using speech to make queries about it illustrates the type of natural experience multimodal interfaces offer to their users. Another key strength of multimodal interfaces is their ability to provide redundancy to accommodate different people and different circumstances. For instance, *MATCHKiosk* [CITATION MJo04 \l 1040] allows one to use speech or handwriting to specify the type of business to search for on a map. Thus, in a noisy setting, one may provide input through handwriting rather than speech. Few other examples of applications of multimodal systems are listed below:

- Smart Video Conferencing
- Intelligent Homes/Offices
- Driver Monitoring
- Intelligent Games
- E-Commerce
- Helping People with Disabilities

In the following sections, some of important applications of multimodal systems have been presented with greater details.

1.9.2 Emotion Recognition Multimodal Systems

As we move towards a world in which computers are more and more ubiquitous, it will become more essential that machines perceive and interpret all clues, implicit and explicit, that we may provide them regarding our intentions. A natural human-computer interaction cannot be based solely on explicitly stated commands. Computers will have to detect the various behavioural signals based on which to infer one's emotional state. This is a significant piece of the puzzle that one has to put together to predict accurately one's intentions and future behaviour.

People are able to make prediction about one's emotional state based on their observations about one's face, body, and voice. Studies show that if one had access to only one of these modalities, the face modality would produce the best predictions. However, this accuracy can be improved by 35% when human judges are given

access to both face and body modalities together. This suggests that affect recognition, which has for the most part focused on facial expressions, can greatly benefit from multimodal fusion techniques. One of the few works that has attempted to integrate more than one modality for affect recognition is *"Automatic prediction of frustration"* [CITATION AKa07 \l 1040] in which facial features and body posture features are combined to produce an indicator of one's frustration.

Another work that integrated face and body modalities is in which the authors showed that, similar to humans, machine classification of emotion is better when based upon face and body data, rather than either modality alone. In *"Analysis of emotion recognition using facial expressions, speech and multimodal information" [CITATION CBu04 \l 1040]*, the authors attempted to fuse facial and voice data for affect recognition. Once again, remaining consistent with human judges, machine classification of emotion as neutral, sad, angry, or happy was most accurate when the facial and vocal data is combined. They recorded the four emotions: "sadness, anger, happiness, and neutral state". The detailed facial motions were captured in conjunctions with simultaneous speech recordings. Deducted experiments showed that the performance of the facial recognition based system overcame the one based on acoustic information only. Results also show that an appropriate fusion of both modalities gave measurable improvements. Results show that the emotion recognition system based on acoustic information only give an overall performance of 70.9 percent, compared to an overall performance of 85 percent for a recognition system based on facial expressions. This is, in fact, due to the fact that the cheek areas give important information for emotion classification. On the other hand, for the bimodal system based on fusing the facial recognition and acoustic information, the overall performance of this classifier was 89.1 percent.

1.10 Map-Based Multimodal Applications

Different input modalities are suitable for expressing different messages. For instance, speech provides an easy and natural mechanism for expressing a query about a selected object or requesting that the object initiate a given operation. However, speech may not be ideal for tasks, such as selection of a particular region on the screen or defining out a particular path. These types of tasks are better accommodated by hand or pen gestures. However, making queries about a given region and selecting that region are all typical tasks that should be accommodate by a map-based interface. Thus, the natural conclusion is that map-based interfaces can greatly improve the user experience by supporting multiple modes of input, especially speech and gestures.

Quickset is one of the more widely known and older map-based applications that make use of speech and pen gesture input. Quickset is a military-training application that allows users to use one of the two modalities or both simultaneously to express a full command. For instance, users may simply draw out with a pen a predefined symbol for platoons at a given location on the map to create a new platoon in that location. Alternatively, users could use speech to specify their intent on creating a new platoon and could specify vocally the co-ordinates in which to place the platoon. Lastly, users could express vocally their intent on making a new platoon while making a pointing gesture with a pen to specify the location of the new platoon. A more recent multimodal map-based application is *Real Hunter*. It is a real-estate interface that expects users to select objects or regions with touch input while making queries using speech. For instance, the user can ask "How much is this?" while pointing to a house on the map.

Tour guides are another type of map-based applications that have shown great potential to benefit from multimodal interfaces. One such example is *MATCHKiosk [CITATION MJo04 \l 1040]*, the interactive city guide. In a similar fashion to *Quickset*, *MATCHKiosk* allows one to express certain queries using speech only, such as “Find me Indian restaurants in Washington.”; using pen input only by circling a region and writing out “restaurants”; using bimodal input by saying “Indian restaurants in this area” and drawing out a circle around Alexandria. These examples illustrate *MATCHKiosk*’s incorporation of handwriting recognition that can frequently substitute for speech input. Although speech may be the more natural option for a user, given the imperfectness of speech, especially in noisy environments, having handwriting as a backup can reduce user frustration.

1.11 Multi-Modal HCI in Medicine

By the early 1980s, surgeons were beginning to reach their limits based on traditional methods alone. Human hand was unfeasible for many tasks and greater magnification and smaller tools were needed. Higher precision was required to localize and manipulate within small and sensitive parts of the human body. Digital robotic neuro-surgery has come as a leading solution to these limitations and emerged fast due to the vast improvements in engineering, computer technology and neuro-imaging techniques. Robotics surgery was introduced into the surgical area [CITATION Uni07 \l 1040]. State University of Aerospace Instrumentation, University of Karlsruhe (Germany) and Harvard Medical School (USA) has been working on developing man-machine interfaces, adaptive robots and multi-agent technologies intended for neuro-surgery. The neuro-surgical robot consists of the following main components: An arm, feedback vision sensors, controllers, a localization system and a data processing centre. Sensors provide the surgeon with feedbacks from the surgical site with real-time imaging, where the latter one updates the controller with new instructions for the robot by using the computer interface and some joysticks.

Neuro-surgical robotics provide the ability to perform surgeries on a much smaller scale with much higher accuracy and precision, giving access to small corridors which is completely important when a brain surgery is involved [CITATION Uni07 \l 1040].

1.12 Conclusion

As technology evolves, the goal of User Experience/HCI is to make it easier for people to use all kinds of digital technologies to accomplish their goals. This includes reducing the barriers to using these tools, whatever the physical ability of the user or the nature of the task; creating a positive user experience including visual and structural elements; ensuring the user can easily navigate the site/device and find what they are looking for; and creating content which is understandable and accessible to the user, especially in a Big Data scenario.

2 BIG DATA SCENARIO

The amount of data in our world has been exploding. Companies capture trillions of bytes of information about their customers, suppliers, and operations, and millions of networked sensors are being embedded in the physical world in devices such as mobile phones and automobiles, sensing, creating, and communicating data. Multimedia and individuals with smartphones and on social network sites will continue to fuel exponential growth. Big data – large pools of data that can be captured, communicated, aggregated, stored, and analyzed – is now part of every sector and function of the global economy. Like other essential factors of production such as hard assets and human capital, it is increasingly the case that much of modern economic activity, innovation, and growth simply couldn't take place without data.

The question is what this phenomenon means. Is the proliferation of data simply evidence of an increasingly intrusive world? Or can big data play a useful economic role? To inform the debate, this study examines the potential value that big data can create for organizations and sectors of the economy and seeks to illustrate and quantify that value, focusing the attention on how is necessary, in the dynamic and fast-changing scenario of big data, to develop *intelligent* algorithms that make the interaction between the end user and the offered service the simplest possible.

2.1 Introduction

Data have become a torrent flowing into every area of the global economy. Companies churn out a burgeoning volume of transactional data, capturing trillions of bytes of information about their customers, suppliers, and operations. millions of networked sensors are being embedded in the physical world in devices such as mobile phones, smart energy meters, automobiles, and industrial machines that sense, create, and communicate data in the age of the Internet of Things. Indeed, as companies and organizations go about their business and interact with individuals, they are generating a tremendous amount of digital "exhaust data," i.e., data that are created as a by-product of other activities. Social media sites, smartphones, and other consumer devices including PCs and laptops have allowed billions of individuals around the world to contribute to the amount of big data available. And the growing volume of multimedia content has played a major role in the exponential growth in the amount of big data. Each second of high-definition video, for example, generates more than 2,000 times as many bytes as required to store a single page of text. In a digitized world, consumers going about their day – communicating, browsing, buying, sharing, searching – create their own enormous trails of data.

2.2 Big Data: Definition

Big data is a relative term describing a situation where the volume, velocity and variety of data exceed an organization's storage or compute capacity for accurate and timely decision making. Some of this data is held in transactional data stores – the byproduct of fast-growing online activity. Machine-to-machine interactions, such as metering, call detail records, environmental sensing and RFID systems, generate their own tidal waves of data. All these forms of data are expanding, and that is coupled with fast-growing streams of unstructured and semistructured data from social media. That's a lot of data, but it is the reality for many organizations. By some estimates, organizations in all sectors have at least 100 terabytes of data, many with more than a petabyte. However, big data is defined less by volume – which is a constantly moving target – than by its ever-increasing variety, velocity, variability and complexity [CITATION SAS12 \l 1040].

• **Variety**. Up to 85 percent of an organization's data is unstructured – not numeric – but it still must be folded into quantitative analysis and decision making. Text, video, audio and other unstructured data require different architecture and technologies for analysis.

• **Velocity**. Initiatives such as the use of RFID tags and smart metering are driving an ever greater need to deal with the torrent of data in nearreal time. This, coupled with the need and drive to be more agile and deliver insight quicker, is putting tremendous pressure on organizations to build the necessary infrastructure and skill base to react quickly enough.

• **Variability**. In addition to the speed at which data comes your way, the data flows can be highly variable – with daily, seasonal and event-triggered peak loads that can be challenging to manage.

• **Complexity**. Difficulties dealing with data increase with the expanding universe of data sources and are compounded by the need to link, match and transform data across business entities and systems.

Organizations need to understand relationships, such as complex hierarchies and data linkages, among all data. A data environment can become extreme along any of the above dimensions or with a

combination of two or all of them at once. However, it is important to understand that not all of your data will be relevant or useful. Organizations must be able to separate the wheat from the chaff and focus on the information that counts – not on the information overload.

2.3 Mapping global Data: Growth and value creation

Many of the most powerful inventions throughout human history, from language to the modern computer, were those that enabled people to better generate, capture, and consume data and information. We have witnessed explosive growth in the amount of data in our world. Big data has reached critical mass in every sector and function of the typical economy, and the rapid development and diffusion of digital information technologies have intensified its growth. We estimate that new data stored by enterprises exceeded 7 exabytes of data globally in 2010 and that new data stored by consumers around the world that year exceeded an additional 6 exabytes.15 To put these very large numbers in context, the data that companies and individuals are producing and storing is equivalent to filling more than 60,000 US Libraries of Congress. If all words spoken by humans were digitized as text, they would total about 5 exabytes – less than the new data stored by consumers in a year.

The increasing volume and detail of information captured by enterprises, together with the rise of multimedia, social media, and the Internet of Things will fuel exponential growth in data for the foreseeable future. There is no doubt that the sheer size and rapidly expanding universe of big data are phenomena in themselves and have been the primary focus of research thus far. But the key question is what broader impact this torrent of data might have. Many consumers are suspicious about the amount of data that is collected about every aspect of their lives, from how they shop to how healthy they are. Is big data simply a sign of how intrusive society has become, or can big data, in fact, play a useful role in economic terms that can benefit all societal stakeholders?

The emphatic answer is that data can indeed create significant value for the world economy, potentially enhancing the productivity and competitiveness of companies and creating a substantial economic surplus for consumers and their governments. Although the relationship between productivity and IT investments is well established, exploring the link between productivity and data breaks new ground. Based on our findings, we believe that the global economy is on the cusp of a new wave of productivity growth enabled by big data.

In this chapter, we look at past and current research on sizing big data and its storage capacity. We then explore the likely relationship between big data and productivity, drawing on past analyses of the impact of IT investment and innovation to drive productivity that we believe is directly applicable to the current and likely future evolution of big data.

Human beings may have limits in their ability to consume and understand Big Data

The generation of big data may be growing exponentially and advancing technology may allow the global economy to store and process ever greater quantities of data, but there may be limits to our innate human ability

– our sensory and cognitive faculties – to process this data torrent. It is said that the mind can handle about seven pieces of information in its short-term memory. Roger Bohn and James Short at the University of California at San Diego discovered that the rate of growth in data consumed by consumers, through various types of media, was a relatively modest 2.8 percent in bytes per hour between 1980 and 2008. We should note that one of the reasons for this slow growth was the relatively fixed number of bytes delivered through television before the widespread adoption of high-definition digital video.2 The topic of information overload has been widely studied by academics from neuroscientists to economists. Economist Herbert Simon once said, *"A wealth of information creates a poverty of attention and a need to allocate that attention efficiently among the overabundance of information sources that might consume it."* Despite these apparent limits, there are ways to help organizations and individuals to process, visualize, and synthesize meaning from big data. For instance, more sophisticated visualization techniques and algorithms, including automated algorithms, can enable people to see patterns in large amounts of data and help them to unearth the most pertinent insights Advancing collaboration technology also allows a large number of individuals, each of whom may possess understanding of a special area of information, to come together in order to create a whole picture to tackle interdisciplinary problems. If organizations and individuals deployed such techniques more widely, end-user demand for big data could strengthen significantly.

2.4 Big Data Techniques and Technologies

A wide variety of techniques and technologies has been developed and adapted to aggregate, manipulate, analyze, and visualize big data. These techniques and technologies draw from several fields including statistics, computer science, applied mathematics, and economics. This means that an organization that intends to derive value from big data has to adopt a flexible, multidisciplinary approach. Some techniques and technologies were developed in a world with access to far smaller volumes and variety in data, but have been successfully adapted so that they are applicable to very large sets of more diverse data. Others have been developed more recently, specifically to capture value from big data. Some were developed by academics and others by companies, especially those with online business models predicated on analyzing big data.

This chapter concentrates on documenting the potential value that leveraging big data can create. It is not a detailed instruction manual on how to capture value, a task that requires highly specific customization to an organization's context, strategy, and capabilities. However, we wanted to note some of the main techniques and technologies that can be applied to harness big data to clarify the way some of the levers for the use of big data that we describe might work. These are not comprehensive lists—the story of big data is still being written; new methods and tools continue to be developed to solve new problems. To help interested readers find a particular technique or technology easily, we have arranged these lists alphabetically. Where we have used bold typefaces, we are illustrating the multiple interconnections between techniques and technologies. We also provide a brief selection of illustrative examples of visualization, a key tool for understanding very large-scale data and complex analyses in order to make better decisions.

2.5 Techniques for analyzing Big Data

There are many techniques that draw on disciplines such as statistics and computer science (particularly machine learning) that can be used to analyze datasets. In this section, we provide a list of some categories of techniques applicable across a range of industries. This list is by no means exhaustive. Indeed, researchers continue to develop new techniques and improve on existing ones, particularly in response to the need to analyze new combinations of data. We note that not all of these techniques strictly require the use of big data – some of them can be applied effectively to smaller datasets (e.g., A/B testing, regression analysis). However, all of the techniques we list here can be applied to big data and, in general, larger and more diverse datasets can be used to generate more numerous and insightful results than smaller, less diverse ones.

A/B Testing

A technique in which a control group is compared with a variety of test groups in order to determine what treatments (i.e., changes) will improve a given objective variable, e.g., marketing response rate. This technique is also known as split testing or bucket testing. An example application is determining what copy text, layouts, images, or colors will improve conversion rates on an e-commerce Web site. Big data enables huge numbers of tests to be executed and analyzed, ensuring that groups are of sufficient size to detect meaningful (i.e., statistically significant) differences between the control and treatment groups. When more than one variable is simultaneously manipulated in the treatment, the multivariate generalization of this technique, which applies statistical modeling, is often called “A/B/N” testing.

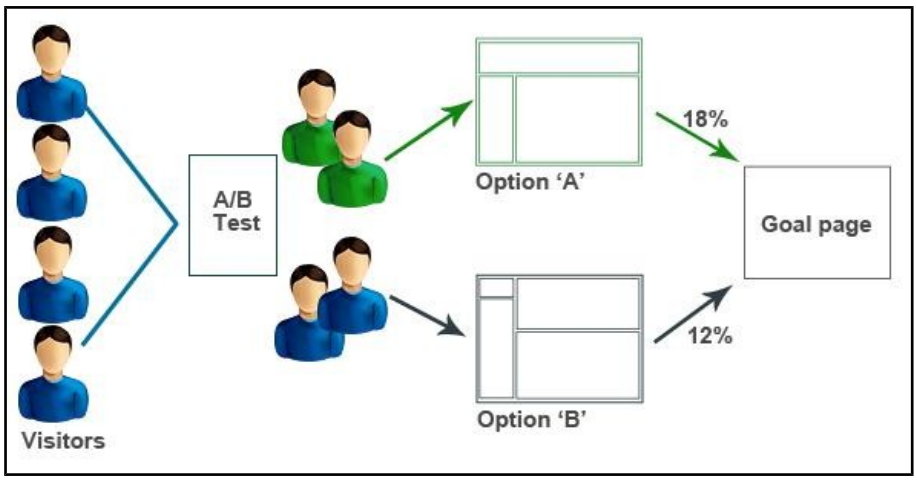


Figure 2-2 A/B Testing in order to split traffic in a website

Association Rule Learning

A set of techniques for discovering interesting relationships, i.e., "association rules," among variables in large databases. These techniques consist of a variety of algorithms to generate and test possible rules. One application is market basket analysis, in which a retailer can determine which products are frequently bought together and use this information for marketing (a commonly cited example is the discovery that many supermarket shoppers who buy diapers also tend to buy beer). Used for data mining.

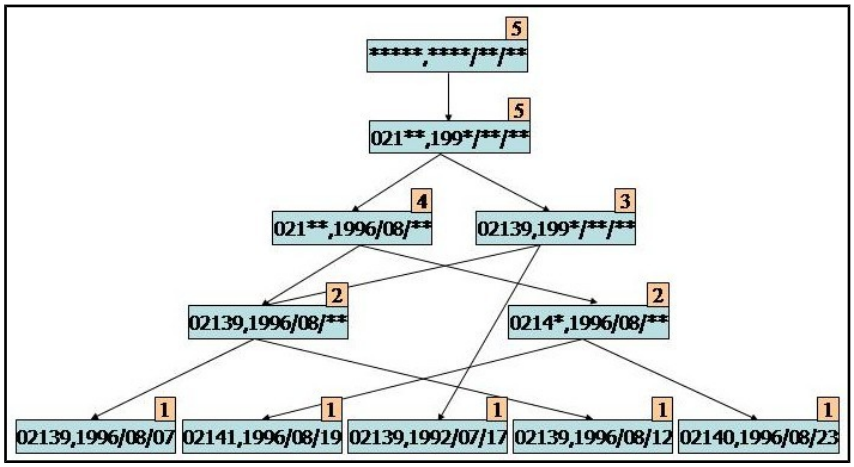


Figure 2-3 Association rule learning example

Classification

A set of techniques to identify the categories in which new data points belong, based on a training set containing data points that have already been categorized. One application is the prediction of segment-specific customer behavior (e.g., buying decisions, churn rate, consumption rate) where there is a clear hypothesis or objective outcome. These techniques are often described as supervised learning because of the existence of a training set; they stand in contrast to cluster analysis, a type of unsupervised learning. Used for data mining.

Cluster Analysis

A statistical method for classifying objects that splits a diverse group into smaller groups of similar objects, whose characteristics of similarity are not known in advance. An example of cluster analysis is segmenting consumers into self-similar groups for targeted marketing. This is a type of unsupervised learning because training data are not used. This technique is in contrast to classification, a type of supervised learning. Used for data mining.

Crowdsourcing

A technique for collecting data submitted by a large group of people or ommunity (i.e., the “crowd”) through an open call, usually through networked media such as the Web.28 This is a type of mass collaboration and an instance of using Web 2.0.

Data Fusion and Data Integration

A set of techniques that integrate and analyze data from multiple sources in order to develop insights in ways that are more efficient and potentially more accurate than if they were developed by analyzing a single source of

data. Signal processing techniques can be used to implement some types of data fusion. One example of an application is sensor data from the Internet of Things being combined to develop an integrated perspective on the performance of a complex distributed system such as an oil refinery. Data from social media, analyzed by natural language processing, can be combined with real-time sales data, in order to determine what effect a marketing campaign is having on customer sentiment and purchasing behavior.

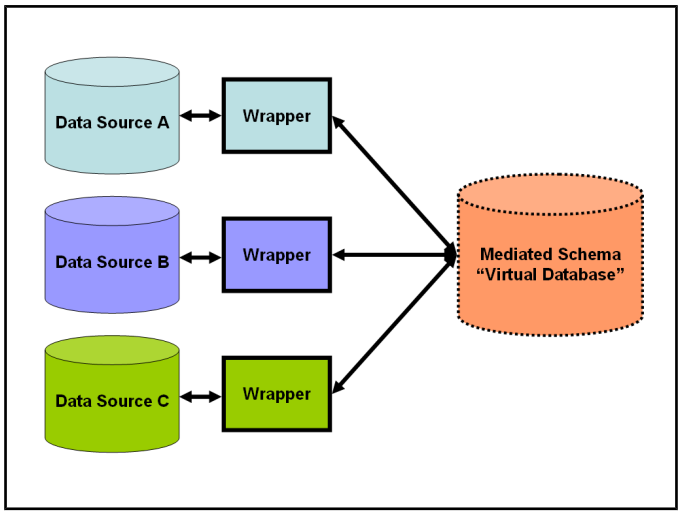


Figure 2-4 Simple schematic for a data-integration solution. A system designer constructs a mediated schema against which users can run queries. The virtual database interfaces with the source databases via wrapper code if required.

Data Mining

A set of techniques to extract patterns from large datasets by combining methods from statistics and machine learning with database management. These techniques include association rule learning, cluster analysis, classification, and regression. Applications include mining customer data to determine segments most likely to respond to an offer, mining human resources data to identify characteristics of most successful employees, or market basket analysis to model the purchase behavior of customers.

Ensemble Learning

Using multiple predictive models (each developed using statistics and/or machine learning) to obtain better predictive performance than could be obtained from any of the constituent models. This is a type of supervised learning.

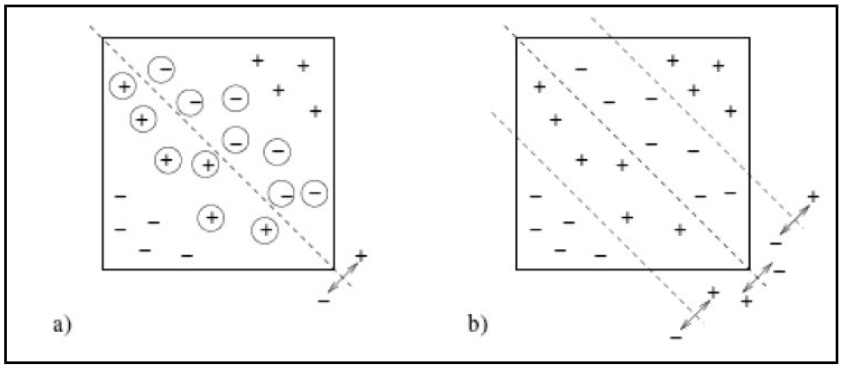


Figure 2-5 Ensemble Learning Algorithms example

Genetic Algorithms

A technique used for optimization that is inspired by the process of natural evolution or "survival of the fittest." In this technique, potential solutions are encoded as "chromosomes" that can combine and mutate. These individual chromosomes are selected for survival within a modeled "environment" that determines the fitness or performance of each individual in the population. Often described as a type of "evolutionary algorithm," these algorithms are well-suited for solving nonlinear problems. Examples of applications include improving job scheduling in manufacturing and optimizing the performance of an investment portfolio.

Machine Learning

A subspecialty of computer science (within a field historically called "artificial intelligence") concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data. A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data. Natural language processing is an example of machine learning.

Natural Language Processing (NLP)

A set of techniques from a subspecialty of computer science (within a field historically called "artificial intelligence") and linguistics that uses computer algorithms to analyze human (natural) language. Many NLP techniques are types of machine learning. One application of NLP is using sentiment analysis on social media to determine how prospective customers are reacting to a branding campaign.

Neural Networks

Computational models, inspired by the structure and workings of biological neural networks (i.e., the cells and connections within a brain), that find patterns in data. Neural networks are well-suited for finding nonlinear patterns. They can be used for pattern recognition and optimization. Some neural network applications involve supervised learning and others involve unsupervised learning. Examples of applications include identifying high-value customers that are at risk of leaving a particular company and identifying fraudulent insurance claims.

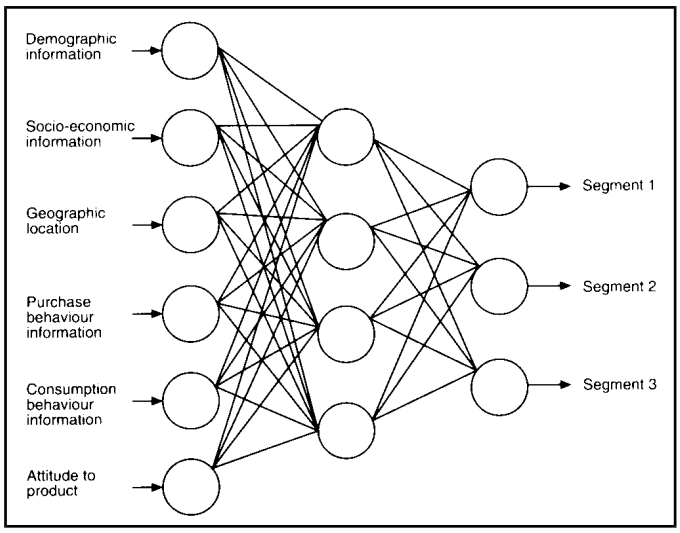


Figure 2-6 Neural Networks and Statistical Techniques in Marketing Research

Network Analysis

A set of techniques used to characterize relationships among discrete nodes in a graph or a network. In social network analysis, connections between individuals in a community or organization are analyzed, e.g., how information travels, or who has the most influence over whom. Examples of applications include identifying key opinion leaders to target for marketing, and identifying bottlenecks in enterprise information flows.

Optimization

A portfolio of numerical techniques used to redesign complex systems and processes to improve their performance according to one or more objective measures (e.g., cost, speed, or reliability). Examples of applications include improving operational processes such as scheduling, routing, and floor layout, and making strategic decisions such as product range strategy, linked investment analysis, and R&D portfolio strategy. Genetic algorithms are an example of an optimization technique.

Pattern Recognition

A set of machine learning techniques that assign some sort of output value (or label) to a given input value (or instance) according to a specific algorithm. Classification techniques are an example.

Predictive Modeling

A set of techniques in which a mathematical model is created or chosen to best predict the probability of an outcome. An example of an application in customer relationship management is the use of predictive models to estimate the likelihood that a customer will “churn” (i.e., change providers) or the likelihood that a customer can be cross-sold another product. Regression is one example of the many predictive modeling techniques.

Regression

A set of statistical techniques to determine how the value of the dependent variable changes when one or more independent variables is modified. Often used for forecasting or prediction. Examples of applications include forecasting sales volumes based on various market and economic variables or determining what measurable manufacturing parameters most influence customer satisfaction. Used for data mining.

Sentiment Analysis

Application of natural language processing and other analytic techniques to identify and extract subjective information from source text material. Key aspects of these analyses include identifying the feature, aspect, or product about which a sentiment is being expressed, and determining the type, "polarity" (i.e., positive, negative, or neutral) and the degree and strength of the sentiment. Examples of applications include companies applying sentiment analysis to analyze social media (e.g., blogs, microblogs, and social networks) to determine how different customer segments and stakeholders are reacting to their products and actions.

Signal Processing

A set of techniques from electrical engineering and applied mathematics originally developed to analyze discrete and continuous signals, i.e., representations of analog physical quantities (even if represented digitally) such as radio signals, sounds, and images. This category includes techniques from signal detection theory, which quantifies the ability to discern between signal and noise. Sample applications include modeling for time series analysis or implementing data fusion to determine a more precise reading by combining data from a set of less precise data sources (i.e., extracting the signal from the noise).

Spatial Analysis

A set of techniques, some applied from statistics, which analyze the topological, geometric, or geographic properties encoded in a data set. Often the data for spatial analysis come from geographic information systems (GIS) that capture data including location information, e.g., addresses or latitude/longitude coordinates. Examples of applications include the incorporation of spatial data into spatial regressions (e.g., how is consumer willingness to purchase a product correlated with location?) or simulations (e.g., how would a manufacturing supply chain network perform with sites in different locations?).

Statistics

The science of the collection, organization, and interpretation of data, including the design of surveys and experiments. Statistical techniques are often used to make judgments about what relationships between variables could have occurred by chance (the "null hypothesis"), and what relationships between variables likely result from some kind of underlying causal relationship (i.e., that are "statistically significant"). Statistical techniques are also used to reduce the likelihood of Type I errors ("false positives") and Type II errors ("false negatives").

An example of an application is A/B testing to determine what types of marketing material will most increase revenue.

Supervised Learning

The set of machine learning techniques that infer a function or relationship from a set of training data. Examples include classification and support vector machines.30 This is different from unsupervised learning.

Simulation

Modeling the behavior of complex systems, often used for forecasting, predicting and scenario planning. Monte Carlo simulations, for example, are a class of algorithms that rely on repeated random sampling, i.e., running thousands of simulations, each based on different assumptions. The result is a histogram that gives a probability distribution of outcomes. One application is assessing the likelihood of meeting financial targets given uncertainties about the success of various initiatives.

Time Series Analysis

Set of techniques from both statistics and signal processing for analyzing sequences of data points, representing values at successive times, to extract meaningful characteristics from the data. Examples of time series analysis include the hourly value of a stock market index or the number of patients diagnosed with a given condition every day. Time series forecasting is the use of a model to predict future values of a time series based on known past values of the same or other series. Some of these techniques, e.g., structural modeling, decompose a series into trend, seasonal, and residual components, which can be useful for identifying cyclical patterns in the data. Examples of applications include forecasting sales figures, or predicting the number of people who will be diagnosed with an infectious disease.

Unsupervised Learning

A set of machine learning techniques that finds hidden structure in unlabeled data. Cluster analysis is an example of unsupervised learning (in contrast to supervised learning). Visualization. Techniques used for creating images, diagrams, or animations to communicate, understand, and improve the results of big data analyses (see the last section of this chapter).

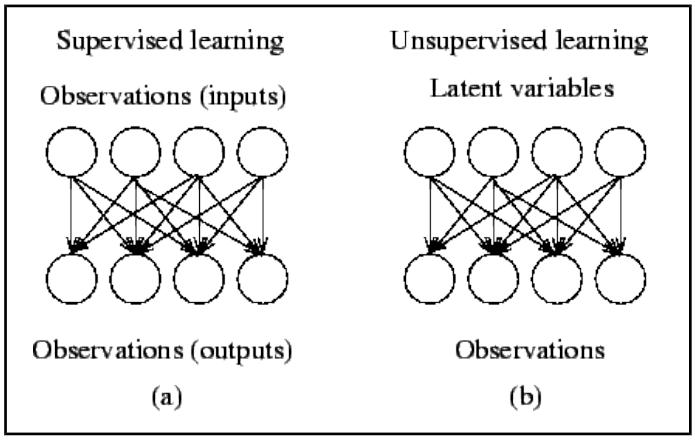


Figure 2-7 Supervised learning vs. Unsupervised learning

2.6 Big Data technologies

There is a growing number of technologies used to aggregate, manipulate, manage, and analyze big data. We have detailed some of the more prominent technologies but this list is not exhaustive, especially as more technologies continue to be developed to support big data techniques, some of which we have listed.

Big Table

Proprietary distributed database system built on the Google File System. Inspiration for HBase.

Business Intelligence (BI)

A type of application software designed to report, analyze, and present data. BI tools are often used to read data that have been previously stored in a data warehouse or data mart. BI tools can also be used to create standard reports that are generated on a periodic basis, or to display information on real-time management dashboards, i.e., integrated displays of metrics that measure the performance of a system.

Cassandra

An open source (free) database management system designed to handle huge amounts of data on a distributed system. This system was originally developed at Facebook and is now managed as a project of the Apache Software foundation.

Cloud Computing

A computing paradigm in which highly scalable computing resources, often configured as a distributed system, are provided as a service through a network.

Data Mart

Subset of a data warehouse, used to provide data to users usually through business intelligence tools.

Data Warehouse

Specialized database optimized for reporting, often used for storing large amounts of structured data. Data is uploaded using ETL (extract, transform, and load) tools from operational data stores, and reports are often generated using business intelligence tools.

Distributed System

Multiple computers, communicating through a network, used to solve a common computational problem. The problem is divided into multiple tasks, each of which is solved by one or more computers working in parallel. Benefits of distributed systems include higher performance at a lower cost (i.e., because a cluster of lower-end computers can be less expensive than a single higher-end computer), higher reliability (i.e., because of a lack of a single point of failure), and more scalability (i.e., because increasing the power of a distributed system can be accomplished by simply adding more nodes rather than completely replacing a central computer).

Dynamo

Proprietary distributed data storage system developed by Amazon. Extract, transform, and load (ETL). Software tools used to extract data from outside sources, transform them to fit operational needs, and load them into a database or data warehouse.

Google File System

Proprietary distributed file system developed by Google; part of the inspiration for Hadoop.

Hadoop

An open source (free) software framework for processing huge datasets on certain kinds of problems on a distributed system. Its development was inspired by Google's MapReduce and Google File System. It was originally developed at Yahoo! and is now managed as a project of the Apache Software Foundation.

HBase

An open source (free), distributed, non-relational database modeled on Google's Big Table. It was originally developed by Powerset and is now managed as a project of the Apache Software foundation as part of the Hadoop.

MapReduce

A software framework introduced by Google for processing huge datasets on certain kinds of problems on a distributed system. Also implemented in Hadoop.

Mashup

An application that uses and combines data presentation or functionality from two or more sources to create new services. These applications are often made available on the Web, and frequently use data accessed through open application programming interfaces or from open data sources.

Metadata

Data that describes the content and context of data files, e.g., means of creation, purpose, time and date of creation, and author.

Non-Relational Database

A database that does not store data in tables (rows and columns). (In contrast to relational database).

R

An open source (free) programming language and software environment for statistical computing and graphics. The R language has become a de facto standard among statisticians for developing statistical software and is widely used for statistical software development and data analysis. R is part of the GNU Project, a collaboration that supports open source projects.

Relational Database

A database made up of a collection of tables (relations), i.e., data are stored in rows and columns. Relational database management systems (RDBMS) store a type of structured data. SQL is the most widely used language for managing relational databases (see item below).

Semi-Structured Data

Data that do not conform to fixed fields but contain tags and other markers to separate data elements. Examples of semi-structured data include XML or HTML-tagged text. Contrast with structured data and unstructured data.

SQL

Originally an acronym for structured query language, SQL is a computer language designed for managing data in relational databases. This technique includes the ability to insert, query, update, and delete data, as well as manage data schema (database structures) and control access to data in the database.

Stream Processing

Technologies designed to process large real-time streams of event data. Stream processing enables applications such as algorithmic trading in financial services, RFID event processing applications, fraud detection, process monitoring, and location-based services in telecommunications. Also known as event stream processing.

Structured Data

Data that reside in fixed fields. Examples of structured data include relational databases or data in spreadsheets. Contrast with semi-structured data and unstructured data.

Unstructured Data

Data that do not reside in fixed fields. Examples include free-form text (e.g., books, articles, body of e-mail messages), untagged audio, image and video data. Contrast with structured data and semi-structured data.

Visualization

Technologies used for creating images, diagrams, or animations to communicate a message that are often used to synthesize the results of big data analyses (see the next section for examples).

2.7 Visualization

Presenting information in such a way that people can consume it effectively is a key challenge that needs to be met if analyzing data is to lead to concrete action. Human beings have evolved to become highly effective at perceiving certain types of patterns with their senses but continue to face significant constraints in their ability to process other types of data such as large amounts of numerical or text data. For this reason, there is a currently a tremendous amount of research and innovation in the field of visualization, i.e., techniques and technologies used for creating images, diagrams, or animations to communicate, understand, and improve the results of big data analyses. We present some examples to provide a glimpse into this burgeoning and important field that supports big data.

2.7.1 Clustergram

A clustergram is a visualization technique used for cluster analysis displaying how individual members of a dataset are assigned to clusters as the number of clusters increases. The choice of the number of clusters is an

important parameter in cluster analysis. This technique enables the analyst to reach a better understanding of how the results of clustering vary with different numbers of clusters.

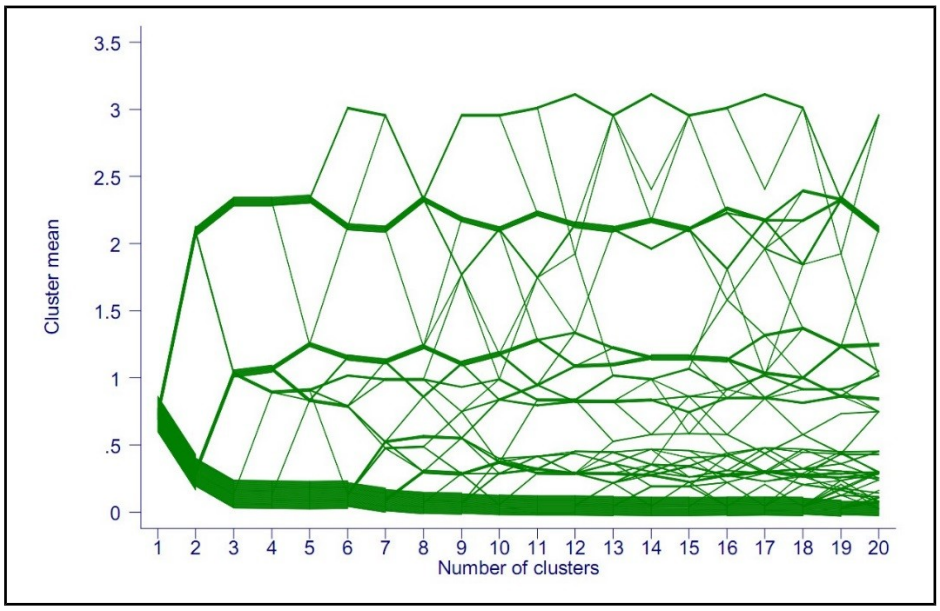


Figure 2-8 Clustergram

2.7.2 History Flow

History flow is a visualization technique that charts the evolution of a document as it is edited by multiple contributing authors. Time appears on the horizontal axis, while contributions to to the text are on the vertical axis; each author has a different color code and the vertical length of a bar indicates the amount of text written by each author. By visualizing the history of a document in this manner, various insights easily emerge. For instance, the history flow we show here that depicts the Wikipedia entry for the word “Islam” shows that an increasing number of authors have made contributions over the history of this entry. One can also see easily that the length of the document has grown over time as more authors have elaborated on the topic, but that, at certain points, there have been significant deletions, too, i.e., when the vertical length has decreased. One can even see

instances of “vandalism” in which the document has been removed completely although, interestingly, the document has tended to be repaired and returned to its previous state very quickly.

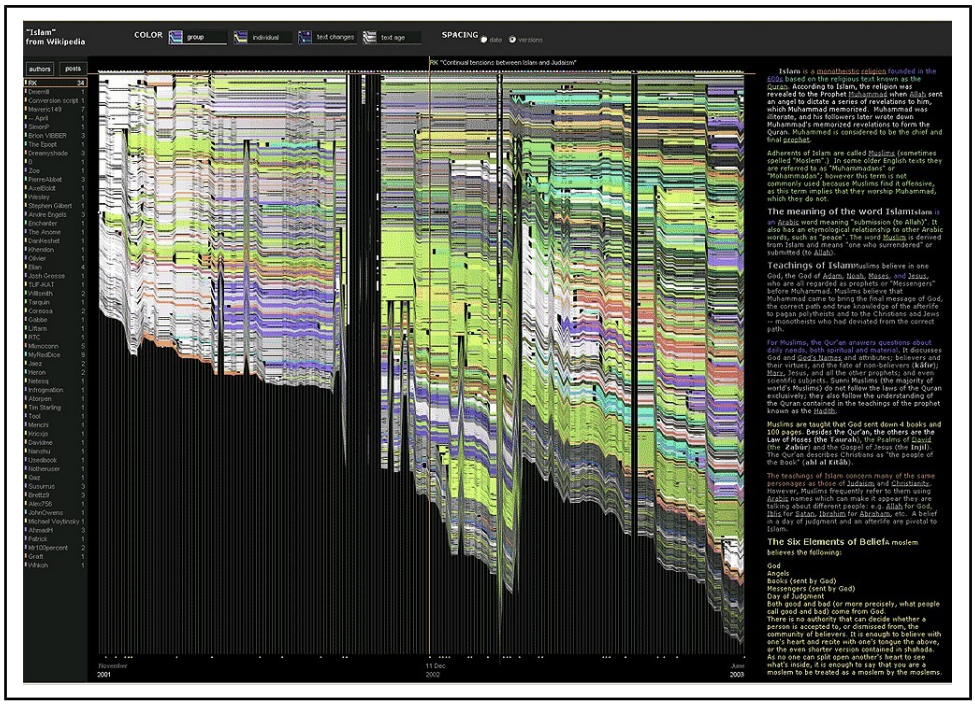


Figure 2-9 History Flow

2.7.3 Spatial Information Flow

Another visualization technique is one that depicts spatial information flows. The example we show here is entitled the New York Talk Exchange. It shows the amount of Internet Protocol (IP) data flowing between New York and cities around the world. The size of the glow on a particular city location corresponds to the amount of IP traffic flowing between that place and New York City; the greater the glow, the larger the flow. This visualization allows us to determine quickly which cities are most closely connected to New York in terms of their communications volume.

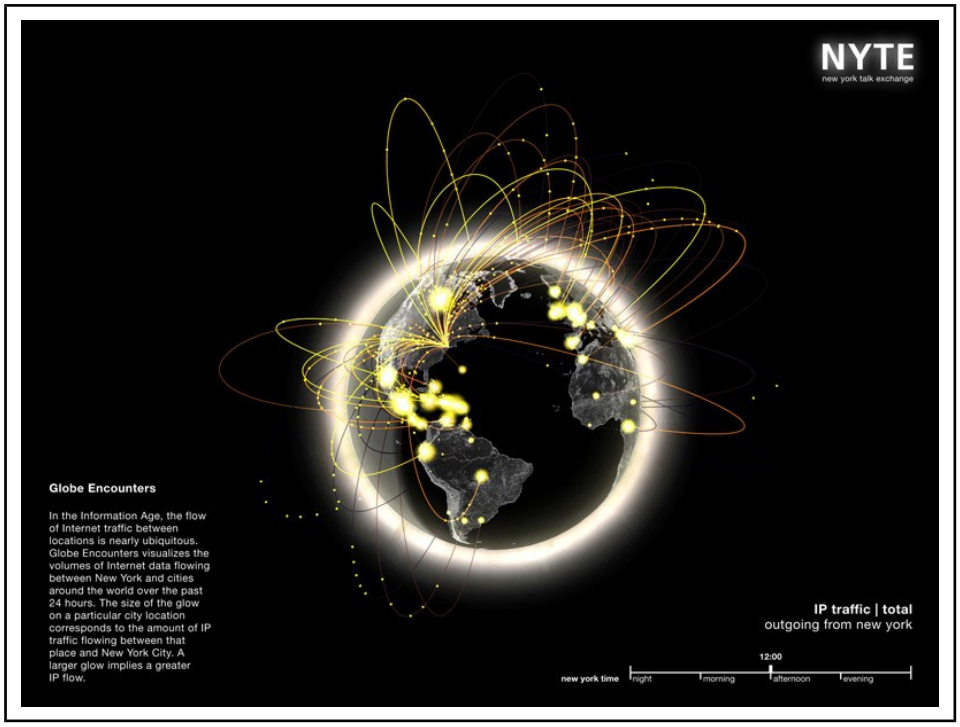


Figure 2-10 Spatial Information Flow

2.8 Challenges and opportunities with Big Data

We are awash in a flood of data today. In a broad range of application areas, data is being collected at unprecedented scale. Decisions that previously were based on guesswork, or on painstakingly constructed models of reality, can now be made based on the data itself. Such Big Data analysis now drives nearly every aspect of our modern society, including mobile services, retail, manufacturing, financial services, life sciences, and physical sciences. Scientific research has been revolutionized by Big Data [CITATION Com11 \l 1040]. The *Sloan Digital Sky Survey* has today become a central resource for astronomers the world over. The field of Astronomy is being transformed from one where taking pictures of the sky was a large part of an astronomer's job to one where the pictures are all in a database already and the astronomer's task is to find interesting objects and phenomena in the database. In the biological sciences, there is now a well-established tradition of depositing scientific data into a public repository, and also of creating public databases for use by other scientists. In fact, there is an entire discipline of bioinformatics that is largely devoted to the curation and analysis of such data. As technology advances, particularly with the advent of Next Generation Sequencing, the size and number of experimental data sets available is increasing exponentially.

Big Data has the potential to revolutionize not just research, but also education [CITATION Com111 \l 1040]. A recent detailed quantitative comparison of different approaches taken by 35 charter schools in NYC has found that one of the top five policies correlated with measurable academic effectiveness was the use of data to guide instruction. Imagine a world in which we have access to a huge database where we collect every detailed measure of every student's academic performance. This data could be used to design the most effective approaches to education, starting from reading, writing, and math, to advanced, college-level, courses. We are far from having access to such data, but there are powerful trends in this direction. In particular, there is a strong trend for massive Web deployment of educational activities, and this will generate an increasingly large amount of detailed data about students' performance. It is widely believed that the use of information technology can reduce the cost of healthcare while improving its quality [CITATION Com112 \l 1040], by making care more preventive and personalized and basing it on more extensive (home-based) continuous monitoring. *McKinsey* [CITATION Jam11 \l 1040] estimates a savings of 300 billion dollars every year in the US alone.

In a similar vein, there have been persuasive cases made for the value of Big Data for urban planning (through fusion of high-fidelity geographical data), intelligent transportation (through analysis and visualization of live and detailed road network data), environmental modeling (through sensor networks ubiquitously collecting data) [CITATION Com113 \l 1040], energy saving (through unveiling patterns of use), smart materials (through the new materials genome initiative), computational social sciences (a new methodology fast growing in popularity because of the dramatically lowered cost of obtaining data), financial systemic risk analysis (through integrated analysis of a web of contracts to find dependencies between financial entities [CITATION Mar11 \l 1040]), homeland security (through analysis of social networks and financial transactions of possible terrorists), computer security (through analysis of logged information and other events, known as *Security Information and Event Management)* and so on.

In 2010, enterprises and users stored more than 13 exabytes of new data; this is over 50,000 times the data in the Library of Congress. The potential value of global personal location data is estimated to be $700 billion to end

users, and it can result in an up to 50% decrease in product development and assembly costs, according to a recent *McKinsey* report [CITATION Jam11 \l 1040]. *McKinsey* predicts an equally great effect of Big Data in employment, where 140,000-190,000 workers with "deep analytical" experience will be needed in the US; furthermore, 1.5 million managers will need to become data-literate. Not surprisingly, the recent PCAST report on Networking and IT R&D identified Big Data as a "research frontier" that can "accelerate progress across a broad range of priorities." Even popular news media now appreciates the value of Big Data as evidenced by coverage in the *Economist*, the *New York Times*, and *National Public Radio*. While the potential benefits of Big Data are real and significant, and some initial successes have already been achieved (such as the Sloan Digital Sky Survey), there remain many technical challenges that must be addressed to fully realize this potential. The sheer size of the data, of course, is a major challenge, and is the one that is most easily recognized. However, there are others. Industry analysis companies like to point out that there are challenges not just in Volume, but also in Variety and Velocity [CITATION Gar11 \l 1040], and that companies should not focus on just the first of these. By Variety, they usually mean heterogeneity of data types, representation, and semantic interpretation. By Velocity, they mean both the rate at which data arrive and the time in which it must be acted upon. While these three are important, this short list fails to include additional important requirements such as privacy and usability.

The analysis of Big Data involves multiple distinct phases as shown in the figure below, each of which introduces challenges. Many people unfortunately focus just on the analysis/modeling phase: while that phase is crucial, it is of little use without the other phases of the data analysis pipeline. Even in the analysis phase, which has received much attention, there are poorly understood complexities in the context of multi-tenanted clusters where several users' programs run concurrently. Many significant challenges extend beyond the analysis phase. For example, Big Data has to be managed in context, which may be noisy, heterogeneous and not include an upfront model. Doing so raises the need to track provenance and to handle uncertainty and error: topics that are crucial to success, and yet rarely mentioned in the same breath as Big Data. Similarly, the questions to the data analysis pipeline will typically not all be laid out in advance. We may need to figure out good questions based on the data. Doing this will require smarter systems and also better support for user interaction with the analysis pipeline. In fact, we currently have a major bottleneck in the number of people empowered to ask questions of the data and analyze it. We can drastically increase this number by supporting many levels of engagement with the data, not all requiring deep database expertise. Solutions to problems such as this will not come from incremental improvements to business as usual such as industry may make on its own. Rather, they require us to fundamentally rethink how we manage data analysis.

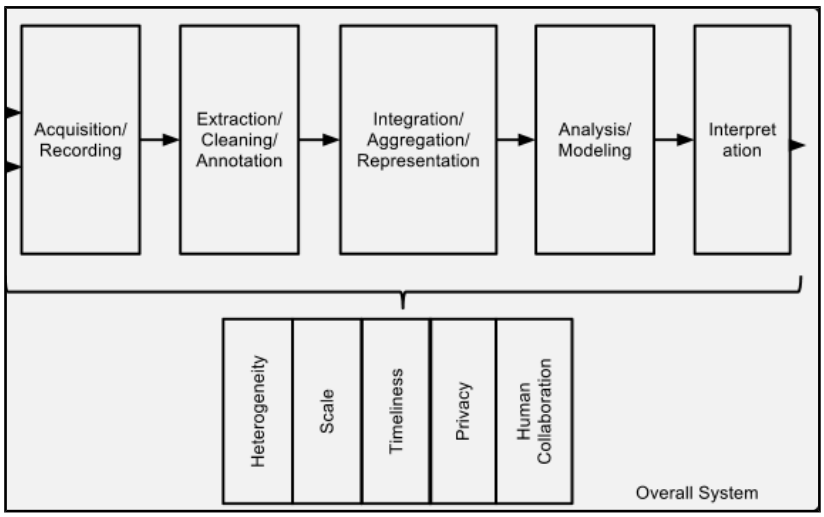


Figure 2-11 The Big Data Analysis Pipeline. Major steps in analysis of Big Data are shown in the flow at top.Below it are big data needs that make these tasks challenging.

Fortunately, existing computational techniques can be applied, either as is or with some extensions, to at least some aspects of the Big Data problem. For example, relational databases rely on the notion of logical data independence: users can think about what they want to compute, while the system (with skilled engineers designing those systems) determines how to compute it efficiently. Similarly, the SQL standard and the relational data model provide a uniform, powerful language to express many query needs and, in principle, allows customers to choose between vendors, increasing competition. The challenge ahead of us is to combine these healthy features of prior systems as we devise novel solutions to the many new challenges of Big Data. In this chapter, we consider each of the boxes in the figure above, and discuss both what has already been done and what challenges remain as we seek to exploit Big Data. We begin by considering the five stages in the pipeline, then move on to the five cross-cutting challenges, and end with a discussion of the architecture of the overall system that combines all these functions.

2.9 Phases in the processing pipeline

2.9.1 Data Acquisition and Recording

Big Data does not arise out of a vacuum: it is recorded from some data generating source. For example, consider our ability to sense and observe the world around us, from the heart rate of an elderly citizen, and presence of toxins in the air we breathe, to the planned square kilometer array telescope, which will produce up to 1 million terabytes of raw data per day. Similarly, scientific experiments and simulations can easily produce petabytes of data today.

Much of this data is of no interest, and it can be filtered and compressed by orders of magnitude. One challenge is to define these filters in such a way that they do not discard useful information. For example, suppose one sensor reading differs substantially from the rest: it is likely to be due to the sensor being faulty, but how can we be sure that it is not an artifact that deserves attention? In addition, the data collected by these sensors most often are spatially and temporally correlated (e.g., traffic sensors on the same road segment). We need research in the science of data reduction that can intelligently process this raw data to a size that its users can handle while not missing the needle in the haystack. Furthermore, we require "on-line" analysis techniques that can process such streaming data on the fly, since we cannot afford to store first and reduce afterward.

The second big challenge is to automatically generate the right metadata to describe what data is recorded and how it is recorded and measured. For example, in scientific experiments, considerable detail regarding specific experimental conditions and procedures may be required to be able to interpret the results correctly, and it is important that such metadata be recorded with observational data. Metadata acquisition systems can minimize the human burden in recording metadata. Another important issue here is data provenance. Recording information about the data at its birth is not useful unless this information can be interpreted and carried along through the data analysis pipeline. For example, a processing error at one step can render subsequent analysis useless; with suitable provenance, we can easily identify all subsequent processing that dependent on this step. Thus we need research both into generating suitable metadata and into data systems that carry the provenance of data and its metadata through data analysis pipelines.

2.9.2 Information Extraction and Cleaning

Frequently, the information collected will not be in a format ready for analysis. For example, consider the collection of electronic health records in a hospital, comprising transcribed dictations from several physicians, structured data from sensors and measurements (possibly with some associated uncertainty), and image data such as x-rays. We cannot leave the data in this form and still effectively analyze it. Rather we require an information extraction process that pulls out the required information from the underlying sources and expresses it in a structured form suitable for analysis. Doing this correctly and completely is a continuing technical challenge.

Note that this data also includes images and will in the future include video; such extraction is often highly application dependent (e.g., what you want to pull out of an MRI is very different from what you would pull out of a picture of the stars, or a surveillance photo). In addition, due to the ubiquity of surveillance cameras and popularity of GPS-enabled mobile phones, cameras, and other portable devices, rich and high fidelity location and trajectory (i.e., movement in space) data can also be extracted. We are used to thinking of Big Data as always telling us the truth, but this is actually far from reality. For example, patients may choose to hide risky behavior and caregivers may sometimes mis-diagnose a condition; patients may also inaccurately recall the name of a drug or even that they ever took it, leading to missing information in (the history portion of) their medical record. Existing work on data cleaning assumes well-recognized constraints on valid data or well-understood error models; for many emerging Big Data domains these do not exist.

2.9.3 Data Integration, Aggregation, and Representation

Given the heterogeneity of the flood of data, it is not enough merely to record it and throw it into a repository. Consider, for example, data from a range of scientific experiments. If we just have a bunch of data sets in a repository, it is unlikely anyone will ever be able to find, let alone reuse, any of this data. With adequate metadata, there is some hope, but even so, challenges will remain due to differences in experimental details and in data record structure. Data analysis is considerably more challenging than simply locating, identifying, understanding, and citing data. For effective large-scale analysis all of this has to happen in a completely automated manner. This requires differences in data structure and semantics to be expressed in forms that are computer understandable, and then "robotically" resolvable. There is a strong body of work in data integration that can provide some of the answers. However, considerable additional work is required to achieve automated error-free difference resolution.

Even for simpler analyses that depend on only one data set, there remains an important question of suitable database design. Usually, there will be many alternative ways in which to store the same information. Certain designs will have advantages over others for certain purposes, and possibly drawbacks for other purposes. Witness, for instance, the tremendous variety in the structure of bioinformatics databases with information regarding substantially similar entities, such as genes. Database design is today an art, and is carefully executed in the enterprise context by highly-paid professionals. We must enable other professionals, such as domain scientists, to create effective database designs, either through devising tools to assist them in the design process or through forgoing the design process completely and developing techniques so that databases can be used effectively in the absence of intelligent database design.

2.9.4 Query Processing, Data Modeling, and Analysis

Methods for querying and mining Big Data are fundamentally different from traditional statistical analysis on small samples. Big Data is often noisy, dynamic, heterogeneous, inter-related and untrustworthy. Nevertheless, even noisy Big Data could be more valuable than tiny samples because general statistics obtained from frequent patterns and correlation analysis usually overpower individual fluctuations and often disclose more reliable hidden patterns and knowledge. Further, interconnected Big Data forms large heterogeneous information networks, with which information redundancy can be explored to compensate for missing data, to crosscheck conflicting cases, to validate trustworthy relationships, to disclose inherent clusters, and to uncover hidden relationships and models.

Mining requires integrated, cleaned, trustworthy, and efficiently accessible data, declarative query and mining interfaces, scalable mining algorithms, and big-data computing environments. At the same time, data mining itself can also be used to help improve the quality and trustworthiness of the data, understand its semantics, and provide intelligent querying functions. As noted previously, real-life medical records have errors, are heterogeneous, and frequently are distributed across multiple systems. The value of Big Data analysis in health care, to take just one example application domain, can only be realized if it can be applied robustly under these difficult conditions. On the flip side, knowledge developed from data can help in correcting errors and removing

ambiguity. For example, a physician may write “DVT” as the diagnosis for a patient. This abbreviation is commonly used for both “deep vein thrombosis” and “diverticulitis,” two very different medical conditions. A knowledge-base constructed from related data can use associated symptoms or medications to determine which of two the physician meant.

Big Data is also enabling the next generation of interactive data analysis with real-time answers. In the future, queries towards Big Data will be automatically generated for content creation on websites, to populate hot-lists or recommendations, and to provide an ad hoc analysis of the value of a data set to decide whether to store or to discard it. Scaling complex query processing techniques to terabytes while enabling interactive response times is a major open research problem today. A problem with current Big Data analysis is the lack of coordination between database systems, which host the data and provide SQL querying, with analytics packages that perform various forms of non-SQL processing, such as data mining and statistical analyses. Today’s analysts are impeded by a tedious process of exporting data from the database, performing a non-SQL process and bringing the data back. This is an obstacle to carrying over the interactive elegance of the first generation of SQL-driven OLAP systems into the data mining type of analysis that is in increasing demand. A tight coupling between declarative query languages and the functions of such packages will benefit both expressiveness and performance of the analysis.

2.9.5 Interpretation

Having the ability to analyze Big Data is of limited value if users cannot understand the analysis. Ultimately, a decision-maker, provided with the result of analysis, has to interpret these results. This interpretation cannot happen in a vacuum. Usually, it involves examining all the assumptions made and retracing the analysis. Furthermore, as we saw above, there are many possible sources of error: computer systems can have bugs, models almost always have assumptions, and results can be based on erroneous data. For all of these reasons, no responsible user will cede authority to the computer system. Rather she will try to understand, and verify, the results produced by the computer. The computer system must make it easy for her to do so. This is particularly a challenge with Big Data due to its complexity. There are often crucial assumptions behind the data recorded. Analytical pipelines can often involve multiple steps, again with assumptions built in. The recent mortgage-related shock to the financial system dramatically underscored the need for such decision-maker diligence – rather than accept the stated solvency of a financial institution at face value, a decision-maker has to examine critically the many assumptions at multiple stages of analysis.

In short, it is rarely enough to provide just the results. Rather, one must provide supplementary information that explains how each result was derived, and based upon precisely what inputs. Such supplementary information is called the provenance of the (result) data. By studying how best to capture, store, and query provenance, in conjunction with techniques to capture adequate metadata, we can create an infrastructure to provide users with the ability both to interpret analytical results obtained and to repeat the analysis with different assumptions, parameters, or data sets.

Systems with a rich palette of visualizations become important in conveying to the users the results of the queries in a way that is best understood in the particular domain. Whereas early business intelligence systems' users were content with tabular presentations, today's analysts need to pack and present results in powerful visualizations that assist interpretation, and support user collaboration as discussed in this chapter. Furthermore, with a few clicks the user should be able to drill down into each piece of data that she sees and understand its provenance, which is a key feature to understanding the data. That is, users need to be able to see not just the results, but also understand why they are seeing those results. However, raw provenance, particularly regarding the phases in the analytics pipeline, is likely to be too technical for many users to grasp completely. One alternative is to enable the users to "play" with the steps in the analysis – make small changes to the pipeline, for example, or modify values for some parameters. The users can then view the results of these incremental changes. By these means, users can develop an intuitive feeling for the analysis and also verify that it performs as expected in corner cases. Accomplishing this requires the system to provide convenient facilities for the user to specify analyses. Declarative specification, discussed in sections, is one component of such a system.

2.10 Challenges in Big Data analysis

Having described the multiple phases in the Big Data analysis pipeline, we now turn to some common challenges that underlie many, and sometimes all, of these phases. These are shown as five boxes in the second row of Fig. 2-12.

2.10.1 Heterogeneity and Incompleteness

When humans consume information, a great deal of heterogeneity is comfortably tolerated. In fact, the nuance and richness of natural language can provide valuable depth. However, machine analysis algorithms expect homogeneous data, and cannot understand nuance. In consequence, data must be carefully structured as a first step in (or prior to) data analysis. Consider, for example, a patient who has multiple medical procedures at a hospital. We could create one record per medical procedure or laboratory test, one record for the entire hospital stay, or one record for all lifetime hospital interactions of this patient. With anything other than the first design, the number of medical procedures and lab tests per record would be different for each patient. The three design choices listed have successively less structure and, conversely, successively greater variety. Greater structure is likely to be required by many (traditional) data analysis systems. However, the less structured design is likely to be more effective for many purposes – for example questions relating to disease progression over time will require an expensive join operation with the first two designs, but can be avoided with the latter. However, computer systems work most efficiently if they can store multiple items that are all identical in size and structure. Efficient representation, access, and analysis of semi-structured data require further work.

Consider an electronic health record database design that has fields for birth date, occupation, and blood type for each patient. What do we do if one or more of these pieces of information is not provided by a patient? Obviously, the health record is still placed in the database, but with the corresponding attribute values being set

to NULL. A data analysis that looks to classify patients by, say, occupation, must take into account patients for which this information is not known. Worse, these patients with unknown occupations can be ignored in the analysis only if we have reason to believe that they are otherwise statistically similar to the patients with known occupation for the analysis performed. For example, if unemployed patients are more likely to hide their employment status, analysis results may be skewed in that it considers a more employed population mix than exists, and hence potentially one that has differences in occupation-related health-profiles. Even after data cleaning and error correction, some incompleteness and some errors in data are likely to remain. This incompleteness and these errors must be managed during data analysis. Doing this correctly is a challenge. Recent work on managing probabilistic data suggests one way to make progress.

2.10.2 Scale

Of course, the first thing anyone thinks of with Big Data is its size. After all, the word “big” is there in the very name. Managing large and rapidly increasing volumes of data has been a challenging issue for many decades. In the past, this challenge was mitigated by processors getting faster, following Moore’s law, to provide us with the resources needed to cope with increasing volumes of data. But, there is a fundamental shift underway now: data volume is scaling faster than compute resources, and CPU speeds are static. First, over the last five years the processor technology has made a dramatic shift - rather than processors doubling their clock cycle frequency every 18-24 months, now, due to power constraints, clock speeds have largely stalled and processors are being built with increasing numbers of cores. In the past, large data processing systems had to worry about parallelism across nodes in a cluster; now, one has to deal with parallelism within a single node.

Unfortunately, parallel data processing techniques that were applied in the past for processing data across nodes don’t directly apply for intra-node parallelism, since the architecture looks very different; for example, there are many more hardware resources such as processor caches and processor memory channels that are shared across cores in a single node. Furthermore, the move towards packing multiple sockets (each with 10s of cores) adds another level of complexity for intra-node parallelism. Finally, with predictions of “dark silicon”, namely that power consideration will likely in the future prohibit us from using all of the hardware in the system continuously, data processing systems will likely have to actively manage the power consumption of the processor. These unprecedented changes require us to rethink how we design, build and operate data processing components. The second dramatic shift that is underway is the move towards cloud computing, which now aggregates multiple disparate workloads with varying performance goals (e.g. interactive services demand that the data processing engine return back an answer within a fixed response time cap) into very large clusters. This level of sharing of resources on expensive and large clusters requires new ways of determining how to run and execute data processing jobs so that we can meet the goals of each workload cost-effectively, and to deal with system failures, which occur more frequently as we operate on larger and larger clusters (that are required to deal with the rapid growth in data volumes). This places a premium on declarative approaches to expressing programs, even those doing complex machine learning tasks, since global optimization across multiple users’ programs is necessary for good overall performance. Reliance on user-driven program optimizations is likely to lead to poor cluster utilization, since users are unaware of other users’ programs. System-driven holistic

optimization requires programs to be sufficiently transparent, e.g., as in relational database systems, where declarative query languages are designed with this in mind.

A third dramatic shift that is underway is the transformative change of the traditional I/O subsystem. For many decades, hard disk drives (HDDs) were used to store persistent data. HDDs had far slower random IO performance than sequential IO performance, and data processing engines formatted their data and designed their query processing methods to "work around" this limitation. But, HDDs are increasingly being replaced by solid state drives today, and other technologies such as Phase Change Memory are around the corner. These newer storage technologies do not have the same large spread in performance between the sequential and random I/O performance, which requires a rethinking of how we design storage subsystems for data processing systems. Implications of this changing storage subsystem potentially touch every aspect of data processing, including query processing algorithms, query scheduling, database design, concurrency control methods and recovery methods.

2.10.3 Timeliness

The flip side of size is speed. The larger the data set to be processed, the longer it will take to analyze. The design of a system that effectively deals with size is likely also to result in a system that can process a given size of data set faster. However, it is not just this speed that is usually meant when one speaks of Velocity in the context of Big Data. Rather, there is an acquisition rate challenge and a timeliness challenge described next. There are many situations in which the result of the analysis is required immediately. For example, if a fraudulent credit card transaction is suspected, it should ideally be flagged before the transaction is completed – potentially preventing the transaction from taking place at all. Obviously, a full analysis of a user's purchase history is not likely to be feasible in real-time. Rather, we need to develop partial results in advance so that a small amount of incremental computation with new data can be used to arrive at a quick determination.

Given a large data set, it is often necessary to find elements in it that meet a specified criterion. In the course of data analysis, this sort of search is likely to occur repeatedly. Scanning the entire data set to find suitable elements is obviously impractical. Rather, index structures are created in advance to permit finding qualifying elements quickly. The problem is that each index structure is designed to support only some classes of criteria. With new analyses desired using Big Data, there are new types of criteria specified, and a need to devise new index structures to support such criteria. For example, consider a traffic management system with information regarding thousands of vehicles and local hot spots on roadways. The system may need to predict potential congestion points along a route chosen by a user, and suggest alternatives. Doing so requires evaluating multiple spatial proximity queries working with the trajectories of moving objects. New index structures are required to support such queries. Designing such structures becomes particularly challenging when the data volume is growing rapidly and the queries have tight response time limits.

2.10.4 Privacy

The privacy of data is another huge concern, and one that increases in the context of Big Data. For electronic health records, there are strict laws governing what can and cannot be done. For other data, regulations, particularly in the US, are less forceful. However, there is great public fear regarding the inappropriate use of personal data, particularly through linking of data from multiple sources. Managing privacy is effectively both a technical and a sociological problem, which must be addressed jointly from both perspectives to realize the promise of big data. Consider, for example, data gleaned from location-based services. These new architectures require a user to share his/her location with the service provider, resulting in obvious privacy concerns. Note that hiding the user's identity alone without hiding her location would not properly address these privacy concerns. An attacker or a (potentially malicious) location-based server can infer the identity of the query source from its (subsequent) location information. For example, a user's location information can be tracked through several stationary connection points (e.g., cell towers). After a while, the user leaves "a trail of packet crumbs" which could be associated to a certain residence or office location and thereby used to determine the user's identity. Several other types of surprisingly private information such as health issues (e.g., presence in a cancer treatment center) or religious preferences (e.g., presence in a church) can also be revealed by just observing anonymous users' movement and usage pattern over time. In general, Barabási et al. showed that there is a close correlation between people's identities and their movement patterns. Note that hiding a user location is much more challenging than hiding his/her identity. This is because with location-based services, the location of the user is needed for a successful data access or data collection, while the identity of the user is not necessary.

There are many additional challenging research problems. For example, we do not know yet how to share private data while limiting disclosure and ensuring sufficient data utility in the shared data. The existing paradigm of differential privacy is a very important step in the right direction, but it unfortunately reduces information content too far in order to be useful in most practical cases. In addition, real data is not static but gets larger and changes over time; none of the prevailing techniques results in any useful content being released in this scenario. Yet another very important direction is to rethink security for information sharing in Big Data use cases. Many online services today require us to share private information (think of Facebook applications), but beyond record-level access control we do not understand what it means to share data, how the shared data can be linked, and how to give users fine-grained control over this sharing.

2.10.5 Human Collaboration

In spite of the tremendous advances made in computational analysis, there remain many patterns that humans can easily detect but computer algorithms have a hard time finding. Indeed, CAPTCHAs exploit precisely this fact to tell human web users apart from computer programs. Ideally, analytics for Big Data will not be all computational – rather it will be designed explicitly to have a human in the loop. The new sub-field of visual analytics is attempting to do this, at least with respect to the modeling and analysis phase in the pipeline. There is similar value to human input at all stages of the analysis pipeline. In today's complex world, it often takes multiple experts from different domains to really understand what is going on. A Big Data analysis system must support input from multiple human experts, and shared exploration of results. These multiple experts may be

separated in space and time when it is too expensive to assemble an entire team together in one room. The data system has to accept this distributed expert input, and support their collaboration.

A popular new method of harnessing human ingenuity to solve problems is through crowd-sourcing. Wikipedia, the online encyclopedia, is perhaps the best known example of crowd-sourced data. We are relying upon information provided by unvetted strangers. Most often, what they say is correct. However, we should expect there to be individuals who have other motives and abilities – some may have a reason to provide false information in an intentional attempt to mislead. While most such errors will be detected and corrected by others in the crowd, we need technologies to facilitate this. We also need a framework to use in analysis of such crowd-sourced data with conflicting statements. As humans, we can look at reviews of a restaurant, some of which are positive and others critical, and come up with a summary assessment based on which we can decide whether to try eating there.

We need computers to be able to do the equivalent. The issues of uncertainty and error become even more pronounced in a specific type of crowd-sourcing, termed participatory-sensing. In this case, every person with a mobile phone can act as a multi-modal sensor collecting various types of data instantaneously (e.g., picture, video, audio, location, time, speed, direction, acceleration). The extra challenge here is the inherent uncertainty of the data collection devices. The fact that collected data are probably spatially and temporally correlated can be exploited to better assess their correctness. When crowd-sourced data is obtained for hire, such as with "Mechanical Turks," much of the data created may be with a primary objective of getting it done quickly rather than correctly. This is yet another error model, which must be planned for explicitly when it applies.

2.10.6 System Architecture

Companies today already use, and appreciate the value of, business intelligence. Business data is analyzed for many purposes: a company may perform system log analytics and social media analytics for risk assessment, customer retention, brand management, and so on. Typically, such varied tasks have been handled by separate systems, even if each system includes common steps of information extraction, data cleaning, relational-like processing (joins, group-by, aggregation), statistical and predictive modeling, and appropriate exploration and visualization tools as shown in Fig. 2-12.

With Big Data, the use of separate systems in this fashion becomes prohibitively expensive given the large size of the data sets. The expense is due not only to the cost of the systems themselves, but also the time to load the data into multiple systems. In consequence, Big Data has made it necessary to run heterogeneous workloads on a single infrastructure that is sufficiently flexible to handle all these workloads. The challenge here is not to build a system that is ideally suited for all processing tasks. Instead, the need is for the underlying system architecture to be flexible enough that the components built on top of it for expressing the various kinds of processing tasks can tune it to efficiently run these different workloads. The effects of scale on the physical architecture were considered in Sec 3.2. In this section, we focus on the programmability requirements.

If users are to compose and build complex analytical pipelines over Big Data, it is essential that they have appropriate high-level primitives to specify their needs in such flexible systems. The Map-Reduce framework

has been tremendously valuable, but is only a first step. Even declarative languages that exploit it, such as Pig Latin, are at a rather low level when it comes to complex analysis tasks. Similar declarative specifications are required at higher levels to meet the programmability and composition needs of these analysis pipelines. Besides the basic technical need, there is a strong business imperative as well. Businesses typically will outsource Big Data processing, or many aspects of it. Declarative specifications are required to enable technically meaningful service level agreements, since the point of the out-sourcing is to specify precisely what task will be performed without going into details of how to do it. Declarative specification is needed not just for the pipeline composition, but also for the individual operations themselves. Each operation (cleaning, extraction, modeling etc.) potentially runs on a very large data set. Furthermore, each operation itself is sufficiently complex that there are many choices and optimizations possible in how it is implemented. In databases, there is considerable work on optimizing individual operations, such as joins. It is well-known that there can be multiple orders of magnitude difference in the cost of two different ways to execute the same query. Fortunately, the user does not have to make this choice – the database system makes it for her. In the case of Big Data, these optimizations may be more complex because not all operations will be I/O intensive as in databases. Some operations may be, but others may be CPU intensive, or a mix. So standard database optimization techniques cannot directly be used. However, it should be possible to develop new techniques for Big Data operations inspired by database techniques.

The very fact that Big Data analysis typically involves multiple phases highlights a challenge that arises routinely in practice: production systems must run complex analytic pipelines, or workflows, at routine intervals, e.g., hourly or daily. New data must be incrementally accounted for, taking into account the results of prior analysis and pre-existing data. And of course, provenance must be preserved, and must include the phases in the analytic pipeline. Current systems offer little to no support for such Big Data pipelines, and this is in itself a challenging objective.

2.11 Conclusion

We have entered an era of Big Data. Through better analysis of the large volumes of data that are becoming available, there is the potential for making faster advances in many scientific disciplines and improving the profitability and success of many enterprises. However, many technical challenges described in this chapter must be addressed before this potential can be realized fully. The challenges include not just the obvious issues of scale, but also heterogeneity, lack of structure, error-handling, privacy, timeliness, provenance, and visualization, at all stages of the analysis pipeline from data acquisition to result interpretation. These technical challenges are common across a large variety of application domains, and therefore not cost-effective to address in the context of one domain alone. Furthermore, these challenges will require transformative solutions, and will not be addressed naturally by the next generation of industrial products. We must support and encourage fundamental research towards addressing these technical challenges if we are to achieve the promised benefits of Big Data.

3 PROBABILISTIC GRAPHICAL MODELS

Research into methods for reasoning under uncertainty is currently one of the most exciting areas of artificial intelligence, largely because it has recently become possible to record, store, and process large amounts of data. While impressive achievements have been made in pattern classification problems such as handwritten character recognition, face detection, speaker identification, and prediction of gene function, it is even more exciting that researchers are on the verge of introducing systems that can perform large-scale combinatorial analyses of data, decomposing the data into interacting components. For example, computational methods for automatic scene analysis are now emerging in the computer vision community. These methods decompose an input image into its constituent objects, lighting conditions, motion patterns, etc.Two of the main challenges are finding effective representations and models in specific applications and finding efficient algorithms for inference and learning in these models. In this paper, we advocate the use of graph-based probability models and their associated inference and learning algorithms. We review exact techniques and various approximate, computationally efficient techniques, including iterated conditional modes, the expectation maximization (EM) algorithm, Gibbs sampling, the mean field method, variational techniques, structured variational techniques and the sum-product algorithm, “loopy” belief propagation. We describe how each technique can be applied in a vision model of multiple, occluding objects and contrast the behaviors and performances of the techniques using a unifying cost function, free energy

3.1 Introduction

A *graphical model* is a type of probabilistic network that has roots in several different research communities, including artificial intelligence (Pearl, 1988), statistics (Lauritzen, 1996), error-control coding (Gallager, 1963), and neural networks. The graphical models framework provides a clean mathematical formalism that has made it possible to understand the relationships among a wide variety of network-based approaches to computation, and in particular to understand many neural network algorithms and architectures as instances of a broader probabilistic methodology.

Graphical models use graphs to represent and manipulate joint probability distributions. The graph underlying a graphical model may be directed, in which case the model is often referred to as a belief network or a Bayesian network [CITATION Jen01 \l 1040] or the graph may be undirected, in which case the model is generally referred to as a Markov random field. A graphical model has both a structural component – encoded by the pattern of edges in the graph – and a parametric component|encoded by numerical “potentials" associated with sets of edges in the graph. The relationship between these components underlies the computational machinery associated with graphical models. In particular, general inference algorithms allow statistical quantities (such as likelihoods and conditional probabilities) and information-theoretic quantities (such as mutual information and conditional entropies) to be computed efficiently. These algorithms are the subject of the current chapter [CITATION Dec96 \l 1040]. Learning algorithms build on these inference algorithms and allow parameters and structures to be estimated from data.

3.2 Background

Directed and undirected graphical models differ in terms of their Markov properties (the relationship between graph separation and conditional independence) and their parameterization (the relationship between local numerical specifications and global joint probabilities). These differences are important in discussions of the family of joint probability distribution that a particular graph can represent. In the inference problem, however, we generally have a specific fixed joint probability distribution at hand, in which case the differences between directed and undirected graphical models are less important. Indeed, in the current chapter, we treat these classes of model together and emphasize their commonalities.

Let U denote a set of nodes of a graph (directed or undirected), and let X_i denote the random variable associated with node i, for $i \in U$. Let X_C denote the subset of random variables associated with a subset of nodes C, for any $C \subseteq U$, and let $X = X_U$ denote the collection of random variables associated with the graph. The family of joint probability distributions associated with a given graph can be parameterized in terms of a product over potential functions associated with subsets of nodes in the graph. For directed graphs, the basic subset on which a potential is dened consists of a single node and its parents, and a potential turns out to be (necessarily) the conditional probability of the node given its parents. Thus, for a directed graph, we have the following representation for the joint probability:

$$p(x) = \prod_i p(x_i \mid x_{\pi_i}), \tag{1}$$

where $p(x_i \mid x_{\pi i})$ is the local conditional probability associated with node i, and π_i is the set of indices labeling the parents of node i. For undirected graphs, the basic subsets are cliques of the graph|subsets of nodes that are completely connected. For a given clique C, let $\psi_C(x_C)$ denote a general potential function|a function that assigns a positive real number to each conFiguretion x_C.

We have:

$$p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C), \tag{2}$$

where C is the set of cliques associated with the graph and Z is an explicit normalizing factor, ensuring that $\sum_x p(x) = 1$. (We work with discrete random variables throughout for simplicity).

Eq. (1) can be viewed as a special case of Eq. (2). Note in particular that we could have included a normalizing factor Z in Eq. (1), but, as is easily verified, it is necessarily equal to one. Second, note that $p(x_i \mid x_{\pi i})$ is a perfectly good example of a potential function, except that the set of nodes that it is defined on – the collection $\{i \cup \pi_i\}$ – is not in general a clique (because the parents of a given node are not in general interconnected). Thus, to treat Eq. (1) and Eq. (2) on an equal footing, we find it convenient to define the so-called *moral graph* G^m associated with a directed graph *G*. The moral graph is an undirected graph obtained by connecting all of the parents of each node in G, and removing the arrowheads. On the moral graph, a conditional probability $p(x_i \mid x_{\pi i})$ is a potential function, and Eq. (1) reduces to a special case of Eq. (2).

3.3 Probablistic Inference

Let (E, F) be a partitioning of the node indices of a graphical model into disjoint subsets, such that (X_E, X_F) is a partitioning of the random variables. There are two basic kinds of inference problem [CITATION Mic \l 1040] that we wish to solve:

- *Marginal probabilities*:

$$p(x_E) = \sum_{x_F} p(x_E, x_F). \tag{3}$$

- *Maximum a posteriori (MAP) probabilities*:

$$p^*(x_E) = \max_{x_F} p(x_E, x_F). \tag{4}$$

From these basic computations we can obtain other quantities of interest. In particular, the conditional probability $p(x_F \mid x_E)$ is equal to:

$$p(x_F \mid x_E) = \frac{p(x_E, x_F)}{\sum_{x_F} p(x_E, x_F)}, \tag{5}$$

and this is readily computed for any x_F once the denominator is computed|a marginalization computation. Moreover, we often wish to combine conditioning and marginalization, or conditioning, marginalization and MAP computations. For example, letting (E, F, H) be a partitioning of the node indices, we may wish to compute:

$$p(x_F \mid x_E) = \frac{p(x_E, x_F)}{\sum_{x_F} p(x_E, x_F)} = \frac{\sum_{x_H} p(x_E, x_F, x_H)}{\sum_{x_F} \sum_{x_H} p(x_E, x_F, x_H)}. \quad (6)$$

We first perform the marginalization operation in the numerator and then perform a subsequent marginalization to obtain the denominator.

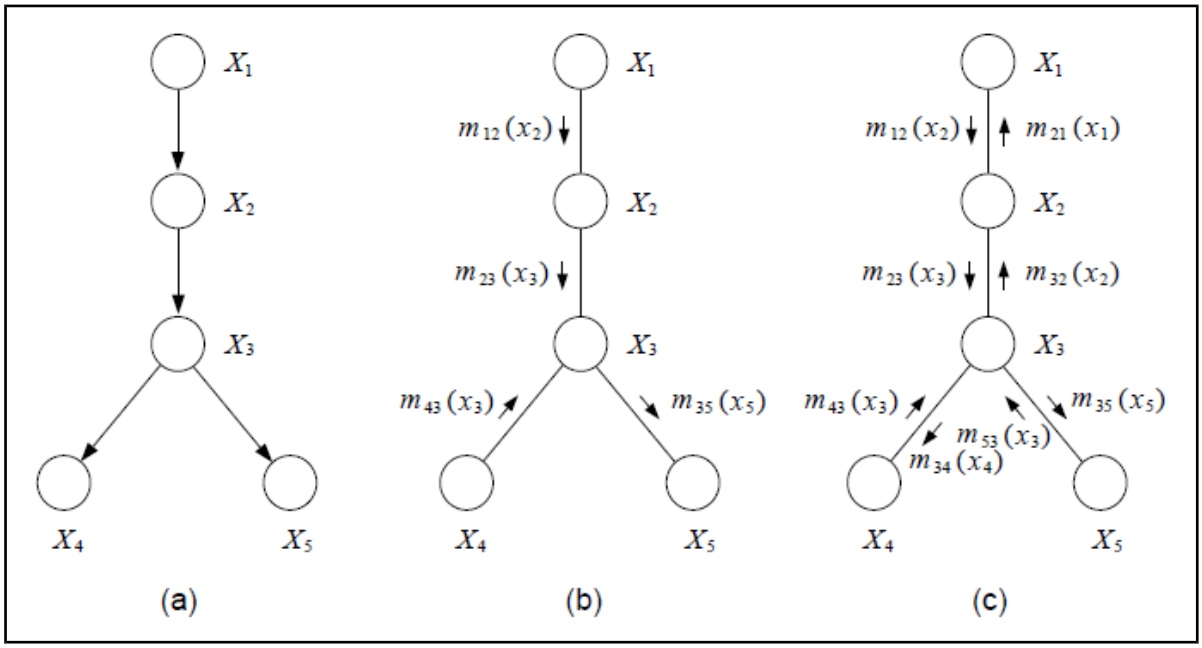


Figure 3-12 (a) A directed graphical model. (b) The intermediate terms that arise during a run of ELIMINATE can be viewed as messages attached to the edges of the moral graph. Here the elimination order was (1, 2, 4, 3). (c) The set of all messages computed by the sum-product algorithm.

3.3.1 Elimination

In this section we introduce a basic algorithm for inference known as *elimination [CITATION Dec96 \l 1040]*. Although elimination applies to arbitrary graphs (as we will see), our focus in this section is on trees. We proceed via an example. Referring to the tree in Figure 3-1a, let us calculate the marginal probability p(x5). We compute this probability by summing the joint probability with respect to $\{x_1, x_2, x_3, x_4\}$. We must pick an order over which to sum, and with some malice of forethought, let us choose the order (1, 2, 4, 3). We have:

$$
\begin{aligned}
p(x_5) &= \sum_{x_3}\sum_{x_4}\sum_{x_2}\sum_{x_1} p(x_1,x_2,x_3,x_4,x_5) \\
&= \sum_{x_3}\sum_{x_4}\sum_{x_2}\sum_{x_1} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)p(x_4 \mid x_3)p(x_5 \mid x_3) \\
&= \sum_{x_3} p(x_5 \mid x_3)\sum_{x_4} p(x_4 \mid x_3)\sum_{x_2} p(x_3 \mid x_2)\sum_{x_1} p(x_1)p(x_2 \mid x_1) \\
&= \sum_{x_3} p(x_5 \mid x_3)\sum_{x_4} p(x_4 \mid x_3)\sum_{x_2} p(x_3 \mid x_2)m_{12}(x_2),
\end{aligned}
$$

where we introduce the notation $m_{ij}(x_j)$ to refer to the intermediate terms that arise in performing the sum. The index i refers to the variable being summed over, and the index j refers to the other variable appearing in the summand (for trees, there will never be more than two variables appearing in any summand). The resulting term is a function of x_j . We continue the derivation:

$$
\begin{aligned}
p(x_5) &= \sum_{x_3} p(x_5 \mid x_3)\sum_{x_4} p(x_4 \mid x_3)\sum_{x_2} p(x_3 \mid x_2)m_{12}(x_2) \\
&= \sum_{x_3} p(x_5 \mid x_3)\sum_{x_4} p(x_4 \mid x_3)m_{23}(x_3) \\
&= \sum_{x_3} p(x_5 \mid x_3)m_{23}(x_3)\sum_{x_4} p(x_4 \mid x_3) \\
&= \sum_{x_3} p(x_5 \mid x_3)m_{23}(x_3)m_{43}(x_3) \\
&= m_{35}(x_5).
\end{aligned}
$$

The final expression is a function of x_5 only and is the desired marginal probability. This computation is formally identically in the case of an undirected graph. In particular, an undirected version of the tree in Figure 3-1a has the parameterization:

$$
p(x) = \frac{1}{Z}\psi_{12}(x_1,x_2)\psi_{23}(x_2,x_3)\psi_{34}(x_3,x_4)\psi_{35}(x_3,x_5). \tag{7}
$$

The first few steps of the computation of $p(x_5)$ are then as follows:

$$p(x_5) = \frac{1}{Z}\sum_{x_3}\psi_{35}(x_3,x_5)\sum_{x_4}\psi_{34}(x_3,x_4)\sum_{x_2}\psi_{23}(x_2,x_3)\sum_{x_1}\psi_{12}(x_1,x_2)$$

and the remainder of the computation proceeds as before.

$$= \frac{1}{Z}\sum_{x_3}\psi_{35}(x_3,x_5)\sum_{x_4}\psi_{34}(x_3,x_4)\sum_{x_2}\psi_{23}(x_2,x_3)m_{12}(x_2),$$

These algebraic manipulations can be summarized succinctly in terms of a general algorithm that we refer to here as Eliminate. The algorithm maintains an *active list* of potentials, which at the outset represent the joint probability and at the end represent the desired marginal probability. Nodes are removed from the graph according to an elimination ordering that must be specified. The algorithm applies to both directed and undirected graphs. Also, as we will see shortly, it is in fact a general algorithm, applying not only to trees but to general graphs.

3.3.2 Message-Passing Algorithms

In many problems we wish to obtain more than a single marginal probability. Thus, for example, we may wish to obtain both $p(x_4)$ and $p(x_5)$ in Figure 3-1(a). Although we could compute each marginal with a separate run of Eliminate, this fails to exploit the fact

```
ELIMINATE(G)
    place all potentials ψ_C(x_C) on the active list
    choose an ordering I of the indices F
    for each X_i in I
        find all potentials on the active list that reference X_i
            and remove them from the active list
        define a new potential as the sum (with respect to x_i) of the
            product of these potentials
        place the new potential on the active list
    end
    return the product of the remaining potentials
```

Figure 3-13 A simple elimination algorithm for marginalization in graphical models.

that common intermediate terms appear in the different runs. We would like to develop an algebra of intermediate terms that allows them to be reused efficiently.

Suppose in particular that we wish to compute $p(x_4)$ in the example in Figure 3-1a. Using the elimination order (1, 2, 5, 3), it is easily verified that we generate the terms $m_{12}(x_2)$ and $m_{23}(x_3)$ as before, and also generate new terms $m_{53}(x_3)$ and $m_{34}(x_4)$.

As suggested by Figure 3-1b, the intermediate terms that arise during elimination can be viewed as messages attached to edges in the moral graph. Rather than viewing inference as an elimination process, based on a global ordering, we instead view inference in terms of local computation and routing of messages. The key operation of summing a product can be written as follows:

$$m_{ij}(x_j) = \sum_{x_i} \psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{ki}(x_i), \tag{8}$$

where N(i) is the set of neighbors of node i. Thus, summing over x_i creates a message $m_{ij}(x_j)$ which is sent to the node j. The reader can verify that each step in our earlier computation of $p(x_5)$ has this form.

A node can send a message to a neighboring node once it has received messages from all of its other neighbors. As in our example, a message arriving at a leaf node is necessarily a marginal probability. In general, the marginal probability at a node is given by the product of all incoming messages:

$$p(x_i) \propto \prod_{k \in N(i)} m_{ki}(x_i). \tag{9}$$

The pair of equations given by Eq. (8) and Eq. (9) define an algorithm known as *sumproduct algorithm* [CITATION Ksc01 \l 1040] or the *belief propagation algorithm [CITATION McE96 \l 1040]*. It is not diffcult to prove that this algorithm is correct for trees. The set of messages needed to compute all of the individual marginal probabilities for the graph in Figure 3-1a is shown in Figure 3-1b.

Note that a pair of messages is sent along each edge, one message in each direction. Neural network also involve message-passing algorithms and local numerical operations. An important difference, however, is that in the neural network setting each node generally has a single activation value that it passes to all of its neighbors. In the sum-product algorithm, on the other hand, individual messages are prepared for each neighbor. Moreover, the message $m_{ij}(x_j)$ from i to j is not included in the product that node j forms in computing a message to send back to node i. The sum-product algorithm avoids *double-counting*.

3.3.3 Maximum A Posteriori (MAP) Probabilities

Referring again to Figure 3-1a, let us suppose that we wish to compute $p^*(x_5)$, the maximum probability conFiguretion of the variables (X_1, X_2, X_3, X_4), for a given value of X_5. Again choosing a particular ordering of the variables, we compute:

$$p^*(x_5) = \max_{x_3} \max_{x_4} \max_{x_2} \max_{x_1} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)p(x_4 \mid x_3)p(x_5 \mid x_3)$$

$$= \max_{x_3} p(x_5 \mid x_3) \max_{x_4} p(x_4 \mid x_3) \max_{x_2} p(x_3 \mid x_2) \max_{x_1} p(x_1)p(x_2 \mid x_1),$$

and the remaining computation proceeds as before. We see that the algebraic operations involved in performing the MAP computation are isomorphic to those in the earlier marginalization computation. Indeed, both the elimination algorithm and the sum-product algorithm extend immediately to MAP computation; we simply replace \sum" with \max" throughout in both cases. The underlying justification is that \max" commutes with products just as sum does.

3.3.4 General Graphs

Our goal in this section is to describe the junction tree algorithm, a generalization of the sum-product algorithm that is correct for arbitrary graphs. We derive the junction tree algorithm by returning to the elimination algorithm. Note that Eliminate is correct for arbitrary graphs|the algorithm simply describes the creation of intermediate terms in a chain of summations that compute a marginal probability. Thus the algorithm is correct, but it is limited to the computation of a single marginal probability.

To show how to generalize the elimination algorithm to allow all individual marginals to be computed, we again proceed by example. Referring to the graph in Figure 3-3a, suppose that we wish to calculate the conditional probability $p(x^1)$. Let us use the elimination ordering (5, 4, 3, 2). At the first step, in which we sum over x5, we remove the potentials

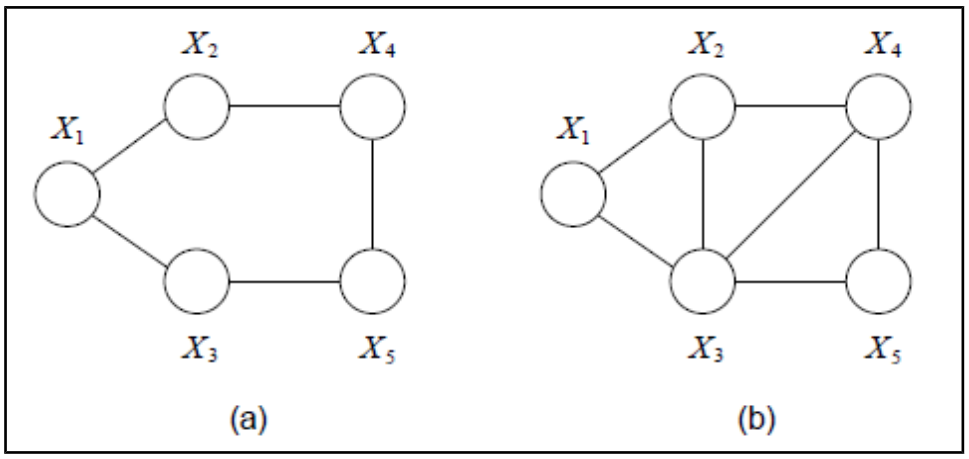


Figure 3-14 (a) An undirected graphical model. (b) The same model, with additional edges that reflect the dependencies created by the elimination algorithm.

$\psi_{35}(x_3, x_5)$ and $\psi_{45}(x_4, x_5)$ from the active list and form the sum:

$$m_{32}(x_3, x_4) = \sum_{x_5} \psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5), \tag{10}$$

where the intermediate term, which is clearly a function of x_3 and x_4, is denoted $m_{32}(x_3, x_4)$. (We explain the subscripts below). The elimination of X_5 has created an intermediate term that effectively links X_3 and X_4, variables that were not linked in the original graph.

Similarly, at the following step, we eliminate X_4:

$$m_{21}(x_2, x_3) = \sum_{x_4} \psi_{24}(x_2, x_4)m_{32}(x_3, x_4) \tag{11}$$

and obtain a term that links X_2 and X_3, variables that were not linked in the original graph.

A graphical record of the dependencies induced during the run of Eliminate is shown in Figure 3-3b.

We could also have created this graph according to a simple graph-theoretic algorithm in which nodes are removed in order from a graph, where, when a node is removed, its remaining neighbors are linked. Thus, for example, when node 5 is removed, nodes 3 and 4 are linked.

When node 4 is removed, nodes 2 and 3 are linked. Let us refer to this algorithm as GraphEliminate.

We can also obtain the desired marginal $p(x_1)$ by working with the filled-in graph in Figure 3-3b from the outset. Noting that the cliques in this graph are $C_1 = \{x_1, x_2, x_3\}$ $C_2 = \{x_2, x_3, x_4\}$ and $C_3 = \{x_3, x_4, x_5\}$, and defining the potentials:

$$\begin{aligned}\psi_{C_1}(x_1, x_2, x_3) &= \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3) \\ \psi_{C_2}(x_2, x_3, x_4) &= \psi_{24}(x_2, x_4) \\ \psi_{C_3}(x_3, x_4, x_5) &= \psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5),\end{aligned}$$

we obtain exactly the same product of potentials as before. Thus we have:

$$p(x) = \frac{1}{Z}\psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3)\psi_{C_2}(x_2, x_3, x_4) = \psi_{24}(x_2, x_4)\psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5)$$

$$\frac{1}{Z}\psi_{C_1}\psi_{C_2}\psi_{C_3}. \tag{12}$$

We have essentially transferred the joint probability distribution from Figure 3-3a to Figure 3-3b. Moreover, the steps of the elimination algorithm applied to Figure 3-2b are exactly the same as before, and we obtain the same marginal. An important difference, however, is that in the case of Figure 3-2b all of the intermediate potentials created during the run of the algorithm are also supported by cliques in the graph. Graphs created by GraphEliminate are known as triangulated graphs, and they have a number of special properties. In particular, they allow the creation of a data structure known as a junction tree on which a generalized message-passing algorithm can be defined.

A junction tree is a tree, in which each node is a clique from the original graph. Messages, which correspond to intermediate terms in Eliminate, pass between these cliques. Although a full discussion of the construction of junction trees is beyond the scope of the article, it is worth noting that a junction tree is not just any tree of cliques from a triangulated graph. Rather, it is a maximal spanning tree (of cliques), with weights given by the cardinalities of the intersections between cliques.

Given a triangulated graph, with cliques $C_i \in \mathcal{C}$ and $\psi_{C_i}(x_{C_i})$, and given a corresponding junction tree (which defines links between the cliques), we send the following message from clique C_i to clique C_j:

$$m_{ij}(x_{S_{ij}}) = \sum_{C_i \setminus S_{ij}} \psi_{C_i}(x_{C_i}) \prod_{k \in \mathcal{N}(i) \setminus j} m_{ki}(x_{S_{ki}}), \tag{13}$$

where $S_{ij} = C_i \cap C_j$, and where N(i) are the neighbors of clique C_i in the junction tree. Moreover, it is possible to prove that we obtain marginal probabilities as products of messages.
Thus:

$$p(x_{C_i}) \propto \prod_{k \in \mathcal{N}(i)} m_{ki}(x_{S_{ki}}) \tag{14}$$

is the marginal probability for clique C_i. (Marginals for single nodes can be obtained via further marginalization: i.e., $p(x_i) = \sum_{C \setminus i} p(x_C)$, for $i \in C$).

The junction tree corresponding to the triangulated graph in Figure 3-2b is shown in Figure 3-4, where the corresponding messages are also shown. The reader can verify that the leftward-going messages are identical to the intermediate terms created during the run of Eliminate. The junction tree algorithm di ers from Eliminate, however, in that messages pass in all directions, and the algorithm yields all clique marginals, not merely thosecorresponding to a single clique.

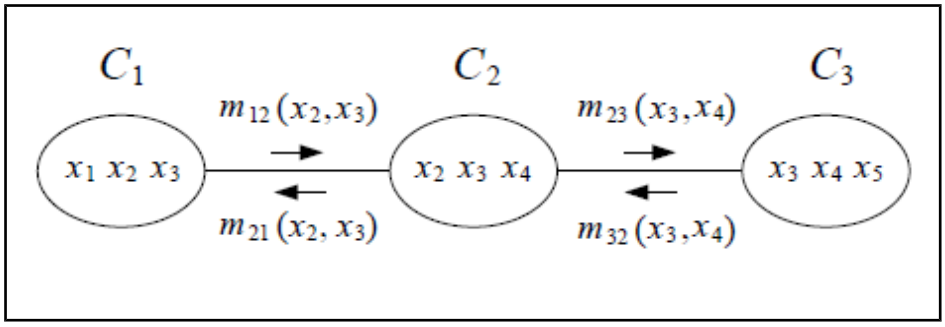


Figure 3-15 The junction tree corresponding to the triangulated graph in Figure 3--3.

The sum-product algorithm described earlier in Eq. (8) and Eq. (9) is a special case of Eq. (13) and Eq. (14), obtained by noting that the original tree in Figure 1(a) is already triangulated, and has pairs of nodes as cliques. In this case, the “separator sets" S_{ij} are singleton nodes.

Once again, the problem of computing MAP probabilities can be solved with a minor change to the basic algorithm. In particular, the \sum" in Eq. (13) is changed to a max.There are many variations on exact inference algorithms, but all of them are either special cases of the junction tree algorithm or are close cousins. The basic message from the research literature on exact inference is that the operations of triangulating a graph and passing messages on the resulting junction tree capture in a succinct way the basic algebraic structure of probabilistic inference.

3.3.5 Computational Complexity

The computational complexity of the junction tree algorithm is a function of the size of the cliques upon which message-passing operations are performed. In particular, summing a clique potential is exponential in the number of nodes in the clique. The problem of finding the optimal triangulation – the triangulation yielding the smallest maximal clique – turns out to be NP-hard. Clearly, if we had to search over all possible elimination orderings, the search would take exponential time. Triangulation can also be defined in other ways, however, and practical triangulation algorithms need not search over orderings. But the problem is still intractable, and can be a practical computational bottleneck. An even more serious problem is that in practical graphical models the original graph may have large cliques, or long loops, and even the optimal triangulation would yield unacceptable complexity. This problem is particularly serious because it arises not during the “compile-time" operation of triangulation, but during the “run-time" operation of messagepassing. Problems in error-control coding and image processing are particularly noteworthy for yielding such graphs, as are discretizations of continuous-time problems, and layered graphs of the kinds studied in the neural network field. To address these problems, we turn

to the topic of approximate probabilistic inference.

3.4 Approximate Inference

The junction tree algorithm focuses on the algebraic structure of probabilistic inference, exploiting the conditional independencies present in a joint probability distribution, as encoded in the pattern of (missing) edges in the graph. There is another form of structure in probability theory, however, that is not exploited in the junction tree framework, and which leads us to hope that successful approximate inference algorithms can be developed. In particular, laws of large numbers and other concentration theorems in probability theory show that sums and products of large numbers of terms can behave in simple, predictable ways, despite the apparent combinatorial complexity of these operations. Approximate algorithms attempt to exploit these numerical aspects of probability theory.

We discuss two large classes of approximate inference algorithms in this section – Monte Carlo algorithms and variational algorithms. While these classes do not exhaust all of the approximation techniques that have been studied, they capture the most widely-used examples.

3.4.1 Monte Carlo Algorithms

Monte Carlo algorithms [CITATION Gil95 \l 1040] are based on the fact that while it may not be feasible to compute expectations under p(x), it may be possible to obtain samples from p(x), or from a closelyrelated distribution, such that marginals and other expectations can be approximated using sample-based averages. We discuss three examples of Monte Carlo algorithms that are commonly used in the graphical model setting –Gibbs

sampling, the Metropolis-Hastings algorithm and importance sampling (for a comprehensive presentation of these methods and others, see Andrieu, et al., in press).

Gibbs sampling is an example of a Markov chain Monte Carlo (MCMC) algorithm [CITATION Gil95 \l 1040]. In an MCMC algorithm, samples are obtained via a Markov chain whose stationary distribution is the desired p(x). The state of the Markov chain is a set of assignments of values to each of the variables, and, after a suitable "burn-in" period so that the chain approaches its stationary distribution, these states are used as samples. The Markov chain for the Gibbs sampler is constructed in a straightforward way: (1) at each step one of the variables X_i is selected (at random or according to some fixed sequence), (2) the conditional distribution $p(x_i \mid x_{U\setminus i})$ is computed, (3) a value x_i is chosen from this distribution, and (4) the sample x_i replaces the previous value of the ith variable. The implementation of Gibbs sampling thus reduces to the computation of the conditional distributions of individual variables given all of the other variables. For graphical models, these conditionals take the following form:

$$p(x_i \mid x_{U\setminus i}) = \frac{\prod_{C\in\mathcal{C}} \psi_C(x_C)}{\sum_{x_i} \prod_{C\in\mathcal{C}} \psi_C(x_C)} = \frac{\prod_{C\in\mathcal{C}_i} \psi_C(x_C)}{\sum_{x_i} \prod_{C\in\mathcal{C}_i} \psi_C(x_C)}, \tag{15}$$

where C_i denotes the set of cliques that contain index i. This set is often much smaller than the set C of all cliques, and in such cases each step of the Gibbs sampler can be implemented efficiently. Indeed, the conditional of node i depends only on the neighbors of node i in the graph, and thus the computation of the conditionals often takes the form of a simple message-passing algorithm that is reminiscent of the sum-product algorithm.

A simple example of a Gibbs sampler is provided by the Boltzmann machine, an undirected graphical model in which the potentials are defined on pairwise cliques. Gibbs sampling is often used for inference in the Boltzmann machine, and the algorithm in Eq. (15) takes the form of the classical computation of the logistic function of a weighted sum of the values of neighboring nodes.

When the computation in Eq. (15) is overly complex, the *Metropolis-Hastings algorithm* can provide an effective alternative. Metropolis-Hastings is an MCMC algorithm [CITATION Gil95 \l 1040][CITATION Adn \l 1040] that is not based on conditional probabilities, and thus does not require normalization. Given the current state x of the algorithm, Metropolis-Hastings chooses a new state ~x from a "proposal distribution," which often simply involves picking a variable X_i at random and choosing a new value for that variable, again at random. The algorithm then computes the "acceptance probability":

$$\alpha = \min\left(1, \frac{\prod_{C\in\mathcal{C}_i} \psi_C(\tilde{x}_C)}{\prod_{C\in\mathcal{C}_i} \psi_C(x_C)}\right). \tag{16}$$

With probability α the algorithm accepts the proposal and moves to ~x, and with probability 1 - α the algorithm remains in state x. For graphical models, this computation also turns out to often take the form of a simple message-passing algorithm.

While Gibbs sampling and Metropolis-Hastings aim at sampling from p(x), importance sampling is a Monte Carlo technique in which samples are chosen from a simpler distribution q(x), and these samples are reweighted appropriately. In particular, we approximate the expectation of a function f(x) as follows:

$$\begin{aligned} E[f(x)] &= \sum_x p(x)f(x) \\ &= \sum_x q(x)\left(\frac{p(x)}{q(x)}f(x)\right) \\ &\approx \frac{1}{N}\sum_{i=1}^{N}\frac{p(x^{(t)})}{q(x^{(t)})}f(x^{(t)}), \end{aligned}$$

where the values $x^{(t)}$ are samples from q(x). The choice of q(x) is in the hands of the designer, and the idea is that q(x) should be chosen to be relatively simple to sample from, while reasonably close to p(x) so that the weight $p(x^{(t)})=q(^{x(t)})$ is reasonably large. In the graphical model setting, natural choices of q(x) are often provided by simplifying the graph underlying p(x) in some way, in particular by deleting edges. The principal advantages of Monte Carlo algorithms are their simplicity of implementation and their generality. Under weak conditions, the algorithms are guaranteed to converge.

A problem with the Monte Carlo approach, however, is that convergence times can be long, and it can be difficult to diagnose convergence. We might hope to be able to improve on Monte Carlo methods in situations in which laws of large numbers are operative. Consider, for example, the case in which a node I has many neighbors, such that the conditional $p(x_i \mid x_{U\setminus i})$ has a single, sharply-determined maximum for most conFiguretions of the neighbors. In this case, it would seem wasteful to continue to sample from this distribution; rather, we would like to be able to compute the maximizing value directly in some way. This way of thinking leads to the variational approach to approximate inference.

3.4.2 Variational Methods

The key to the variational approach lies in converting the probabilistic inference problem into an optimization problem, such that the standard tools of constrained optimization can be exploited. The basic approach has a similar flavor to importance sampling, but instead of choosing a single q(x) a priori, a family of approximating distributions f{(x)} is used, and the optimization machinery chooses a particular member from this family.

We begin by showing that the joint probability p(x) can be viewed as the solution toan optimization problem. In particular, define the energy of a conFiguretion x by E(x) = - log p(x) - log Z, and define the variational free energy as follows:

$$\begin{aligned} F(\{q(x)\}) &= \sum_x q(x)E(x) + \sum_x q(x)\log q(x) \\ &= -\sum_x q(x)\log p(x) + \sum_x q(x)\log q(x) - \log Z. \end{aligned}$$

The variational free energy is equal (up to an additive constant) to the Kullback-Leibler divergence between q(x) and p(x). It is therefore minimized when q(x) = p(x) and attains the value of - log Z at the minimum. We have thus characterized p(x) variationally. Minimizing F is as diffcult as doing exact inference, and much effort has been invested into finding approximate forms of F that are easier to minimize. Each approximate version of F gives a approximate variational inference algorithm.

For example, the simplest variational algorithm is the mean field approximation, in which {q(x)} is restricted to the family of factorized distributions: $q(x) = \prod_i q_i(x_i)$. In this case F simplifies to:

$$F_{MF}(\{q_i\}) = -\sum_C \sum_{x_C} \log \psi_C(x_C) \prod_{i \in C} q_i(x_i) + \sum_i \sum_{x_i} q_i(x_i) \log q_i(x_i), \tag{17}$$

subject to the constraint $\sum_{x_i} q_i(x_i) = 1$.
Setting the derivative with respect to $q_i(x_i)$ equal to zero gives:

$$q_i(x_i) = \alpha \exp\left(\sum_C \sum_{x_C \setminus i} \log \psi_C(x_C) \prod_{j \in C, j \neq i} q_j(x_j) \right) \tag{18}$$

where α is a normalization constant chosen so that $\sum_{x_i} q_i(x_i) = 1$. The sum over cliques C is over all cliques that node i belongs to.

Eq. (18) defines an approximate inference algorithm. We initialize approximate marginal $q(x_i)$ for all nodes in the graph and then update the approximate marginal at one node based on those at neighboring nodes (note that the right hand side of Eq. (18) depends only on cliques that node i belongs to). This yields a message-passing algorithm that is similar to neural network algorithms; in particular, the value $q(x_i)$ can be viewed as the “activation" of node i.

More elaborate approximations to the free energy give better approximate marginal probabilities. While the mean field free energy depends only on approximate marginals at single nodes, the Bethe free energy depends on approximate marginals at single nodes $q_i(x_i)$ as well as approximate marginals on cliques $q_C(x_C)$:

$$F_\beta(\{q_i, q_C\}) = \sum_C \sum_{x_C} q_C(x_C) \log \frac{q_C(x_C)}{\psi_c(x_C)} \tag{19}$$

$$- \sum_i (d_i - 1) \sum_{x_i} q_i(x_i) \log q_i(x_i) \tag{20}$$

where d_i - 1 denotes the number of cliques that node i belongs to.

The approximate clique marginals and the approximate singleton marginals must satisfy a simple marginalization constraint: $\sum_{x_C \setminus i} q_C(x_C) = q_i(x_i)$. When we add Lagrange multipliers and differentiate the Lagrangian we obtain a set of fixed point equations. Surprisingly, these equations end up being equivalent to the \sum-product" algorithm for trees in Eq. (8). The messages mij(xj) are simply exponentiated Lagrange multipliers.

Thus the Bethe approximation is equivalent to applying the local message-passing scheme developed for trees to graphs that have loops. This approach to approximate inference has been very successful in the domain of error-control coding, allowing practical codes based on graphical models to nearly reach the Shannon limit.

4 BAYESIAN NETWORKS

Bayesian networks have been receiving considerable attention over the last few decades from scientists and engineers across a number of fields, including computer science, cognitive science, statistics, and philosophy [CITATION Adn09 \l 1040]. In computer science, the development of Bayesian networks was driven by research in artificial intelligence, which aimed at producing a practical framework for commonsense reasoning. Statisticians have also contributed to the development of Bayesian networks, where they are studied under the broader umbrella of probabilistic graphical models. Interestingly enough, a number of other more specialized fields, such as genetic linkage analysis, speech recognition, information theory and reliability analysis, have developed representations that can be thought of as concrete instantiations or restricted cases of Bayesian networks. For example, pedigrees and their associated phenotype/genotype information, reliability

block diagrams, and hidden Markov models (used in many fields including speech recognition and bioinformatics) can all be viewed as Bayesian networks. Canonical instances of Bayesian networks also exist and have been used to solve standard problems that span across domains

such as computer vision, the Web, and medical diagnosis.

4.1 Introduction

Bayesian networks provide a systematic and localized method for structuring probabilistic information about a situation into a coherent whole. They also provide a suite of algorithms that allow one to automatically derive many implications of this information, which can form the basis for important conclusions and decisions about the corresponding situation (for example, computing the overall reliability of a system, finding the most likely message that was sent across a noisy channel, identifying the most likely users that would respond to an ad, restoring a noisy image, mapping genes onto a chromosome, among others). Technically speaking, a Bayesian network is a compact representation of a probability distribution that is usually too large to be handled using traditional specifications from probability and statistics such as tables and equations [CITATION Adn \l 1040].

For example, Bayesian networks with thousands of variables have been constructed and reasoned about successfully, allowing one to efficiently represent and reason about probability distributions whose size is exponential in that number of variables (for example, in genetic link-age analysis, low-level vision, and networks synthesized from relational models). Every Bayesian network has two components: a directed acyclic graph (called a structure), and a set of conditional probability tables (CPTs). The nodes of a structure correspond to the variables of interest, and its edges have a formal interpretation in terms of probabilistic independence. We will discuss this interpretation later, but suffice to say here that in many practical applications, one can often interpret network edges as signifying direct causal influences. A Bayesian network must include a CPT for each variable, which quantifies the relationship between that variable and its parents in the network. For example, the CPT for variable A specifies the conditional probability distribution of A given its parents F and T. According to this CPT, the probability of A = true given F = true and T = false is Pr(A=true|F = true; T = false) = .9900 and is called a network parameter.a A main feature of Bayesian networks is their guaranteed consistency and completeness as there is one and only one probability distribution that satisfies the constraints of a Bayesian network.

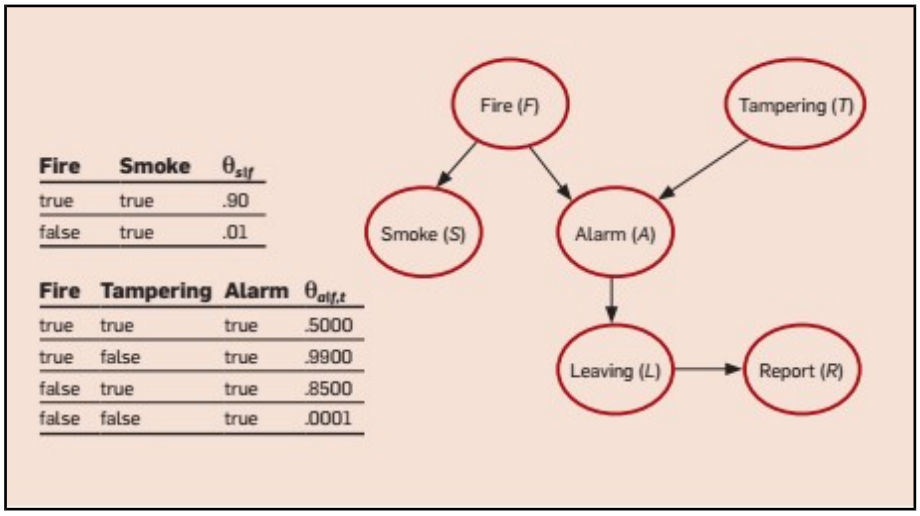


Figure 4-16 A Bayesian network with some of its conditional probability tables (CPTs).

For example, the network in Figure 4-1 induces a unique probability distribution over the 64 instantiations of its variables. This distribution provides enough information to attribute a probability to every event that can be expressed using the variables appearing in this network, for example, the probability of alarm tampering given no smoke and a report of people leaving the building.

Another feature of Bayesian networks is the existence of efficient algorithms for computing such probabilities without having to explicitly generate the underlying probability distribution (which would be computationally infeasible for many interesting networks). These algorithms, to be discussed in detail later, apply to any Bayesian network, regardless of its topology. Yet, the efficiency of these algorithms – and their accuracy in the case of approximation algorithms – may be quite sensitive to this topology and the specific query at hand. Interestingly enough, in domains such as genetics, reliability analysis, and information theory, scientists have developed algorithms that are indeed subsumed by the more general algorithms for Bayesian networks. In fact, one of the main objectives of this article is to raise awareness about these connections. The more general objective, however, is to provide an accessible introduction to Bayesian networks, which allows scientists and engineers to more easily identify problems that can be reduced to Bayesian network inference, putting them in a position where they can capitalize on the vast progress that has been made in this area over the last few decades.

4.2 Casuality and indipendence

We will start by unveiling the central insight behind Bayesian networks that allows them to compactly represent very large distributions. Consider Figure 4-1 and the associated CPTs. Each probability that appears in one of these CPTs does specify a constraint that must be satisfied by the distribution induced by the network. For

example, the distribution must assign the probability .01 to having smoke without fire, Pr(S = true|F = false), since this is specified by the CPT of variable S. These constraints, however,

are not sufficient to pin down a unique probability distribution. So what additional information is being appealed to here? The answer lies in the structure of a Bayesian network, which specifies additional constraints in the form of probabilistic conditional independencies.

In particular, every variable in the structure is assumed to become independent of its non-descendants once its parents are known. In Figure 4-1, variable L is assumed to become independent of its non-descendants T, F, S once its parent A is known. In other words, once the value of variable A is known, the probability distribution of variable L will no longer change due to new information about variables T, F and S. Another example from Figure 4-1: variable A is assumed to become independent of its non-descendant S once its parents F and T are known. These independence constraints are known as the Markovian assumptions of a Bayesian network. Together with the numerical constraints specified by CPTs, they are satisfied by exactly one probability distribution. Does this mean that every time a Bayesian network is constructed, one must verify the conditional independencies asserted by its structure? This really depends on the construction method. I will discuss three main methods in the section entitled "How Are Bayesian Networks Constructed?" that include subjective construction, synthesis from other specifications, and learning from data.

The first method is the least systematic, but even in that case, one rarely thinks about conditional independence when constructing networks. Instead, one thinks about causality, adding the edge $X \rightarrow Y$ whenever X is perceived to be a direct cause of Y. This leads to a causal structure in which the Markovian assumptions read: each variable becomes independent of its non-effects once its direct causes are known. The ubiquity of Bayesian networks stems from the fact that people are quite good at identifying direct causes from a given set of variables, and at deciding whether the set of variables contains all of the relevant direct causes. This ability is all that one needs for constructing a causal structure. The distribution induced by a Bayesian network typically satisfies additional independencies, beyond the Markovian ones discussed above. Moreover, all such independencies can be identified efficiently using a graphical test known as d-separation. According to this test, variables X and Y are guaranteed to be independent given variables Z if every path between X and Y is blocked by Z. Intuitively, a path is blocked when it cannot be used to justify a dependence between X and Y in light of our knowledge of Z.

For an example, consider the path $\alpha : S \leftarrow F \rightarrow A \leftarrow T$ in Figure 4-1 and suppose we know the alarm has triggered (that is, we know the value of variable A). This path can then be used to establish a dependence between variables S and T as follows. First, observing smoke increases the likelihood of fire since fire is a direct cause of smoke according to path α. Moreover, the increased likelihood of fire explains away tampering as a cause of the alarm, leading to a decrease in the probability of tampering (fire and tampering are two competing causes of the alarm according to path α). Hence, the path could be used to establish a dependence between S and T in this case. Variables S and T are therefore not independent given A due to the presence of this unblocked path. One can verify, however, that this path cannot be used to establish a dependence between S and T in case we know the value of variable F instead of A. Hence, the path is blocked by F. Even though we appealed to the notion of causality when describing the d-separation test, one can phrase and prove the test without any appeal to causality—we only need the Markovian assumptions. The full d-separation test gives the precise conditions under which a path between two variables is blocked, guaranteeing independence whenever all paths are blocked. The test can be implemented in time linear in the Bayesian network structure, without the need to

explicitly enumerate paths as suggested previously. The d-separation test can be used to directly derive results that have been proven for specialized probabilistic models used in a variety of fields.

One example is hidden Markov models (HMMs), which are used to model dynamic systems whose states are not observable, yet their outputs are. One uses an HMM when interested in making inferences about these changing states, given the sequence of outputs they generate. HMMs are widely used in applications requiring temporal pattern recognition, including speech, handwriting, and gesture recognition; and various problems in bioinformatics.

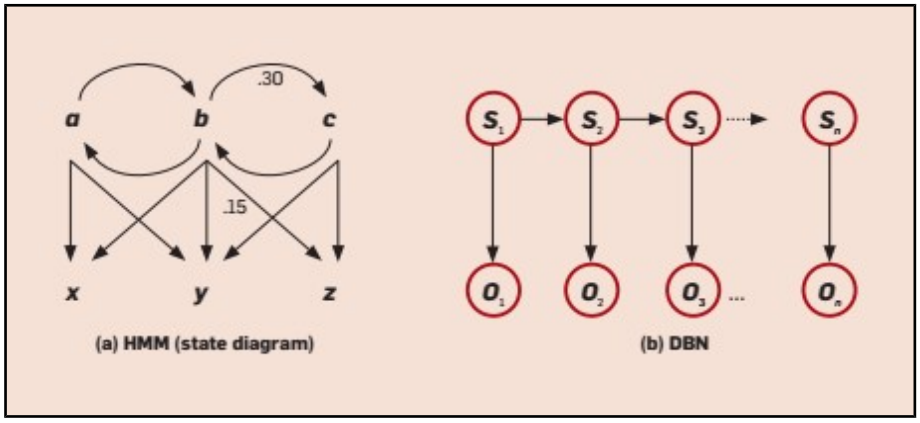


Figure 4-17 A Hidden Markov Model (HMM) and its corresponding Dynamic Bayesian Network (DBN).

Figure 4-2a depicts an HMM, which models a system with three states (a, b, c) and three outputs (x, y, z). The figure depicts the possible transitions between the system states, which need to be annotated by their probabilities. For example, state b can transition to states a or c, with a 30% chance of transitioning to state c. Each state can emit a number of observable outputs, again, with some probabilities.

In this example, state b can emit any of the three outputs, with output z having a 15% chance of being emitted by this state. This HMM can be represented by the Bayesian network in Figure 4-2b. Here, variable St has three values a, b, c and represents the system state at time t, while variable Ot has the values x, y, z and represents the system output at time t. Using d-separation on this network, one can immediately derive the characteristic property of HMMs: once the state of the system at time t is known, its states and outputs at times > t become independent of its states and outputs at times < t.

We also note the network in Figure 4-2b is one of the simplest instances of what is known as dynamic Bayesian networks (DBNs). A number of extensions have been considered for HMMs, which can be viewed as more structured instances of DBNs. When proposing such extensions, however, one has the obligation of offering a corresponding algorithmic toolbox for inference. By viewing these extended HMMs as instances of Bayesian networks, however, one immediately inherits the corresponding Bayesian network algorithms for this purpose.

4.3 How are Bayesian Networks constructed?

One can identify three main methods for constructing Bayesian networks. According to the first method, which is largely subjective, one reflects on their own knowledge or the knowledge of others (typically, perceptions about causal influences) and then captures them into a Bayesian network. The network in Figure 4-1 is an example of this construction method.

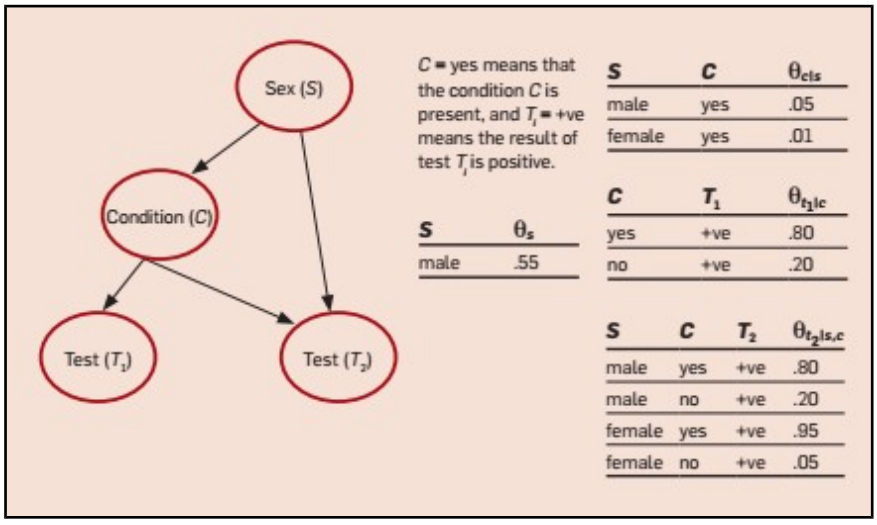


Figure 4-18 A Bayesian network that models a population, a medical condition, and two corresponding tests.

The network structure of Figure 4-3 depicts another example, yet the parameters of this network can be obtained from more formal sources, such as population statistics and test specifications. According to this network, we have a population that is 55% males and 45% females, whose members can suffer from a medical condition C that is more likely to occur in males.

Moreover, two diagnostic tests are available for detecting this condition, T_1 and T_2, with the second test being more effective on females. The CPTs of this network also reveal that the two tests are equally effective on males. The second method for constructing Bayesian networks is based on automatically synthesizing them from some other type of formal knowledge. For example, in many applications that involve system analysis, such as reliability and diagnosis, one can synthesize a Bayesian network automatically from a formal system design.

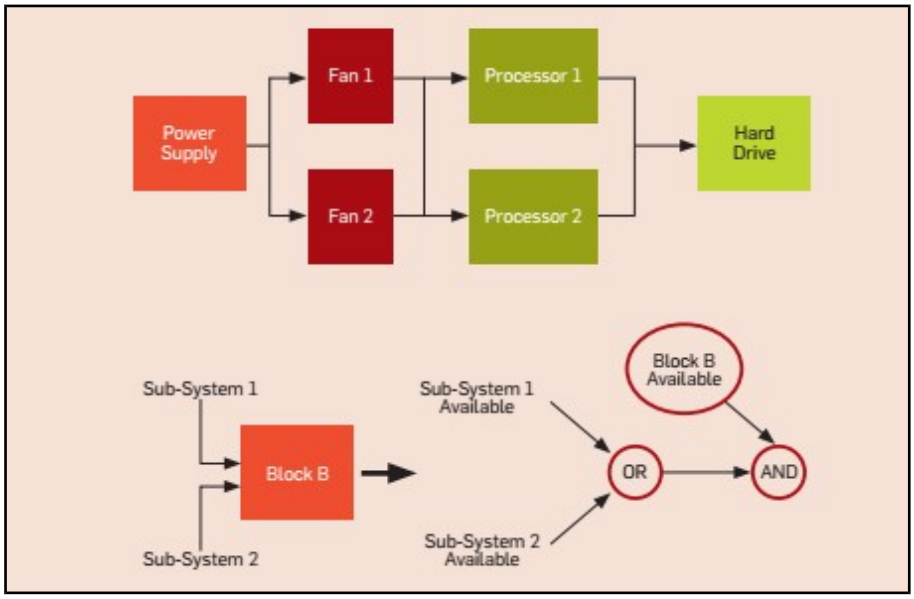


Figure 4-19 A reliability block diagram (top), with a systematic method for mapping its blocks into Bayesian network fragments (bottom).

Figure 4-4 depicts a reliability block diagram (RBD) used in reliability analysis. The RBD depicts system components and the dependencies between their availability. For example, Processor 1 requires either of the fans for its availability, and each of the fans requires power for its availability. The goal here is to compute the overall reliability of the system (probability of its availability) given the reliabilities of each of its components. Figure 4-4 shows also how one may systematically convert each block in an RBD into a Bayesian network fragment, while Figure 4-3 depicts the corresponding Bayesian network constructed according to this method. The CPTs of this figure can be completely constructed based on the reliabilities of individual components (not shown here) and the semantics of the transformation method. The third method for constructing Bayesian networks is based on learning them from data, such as medical records or student admissions data.

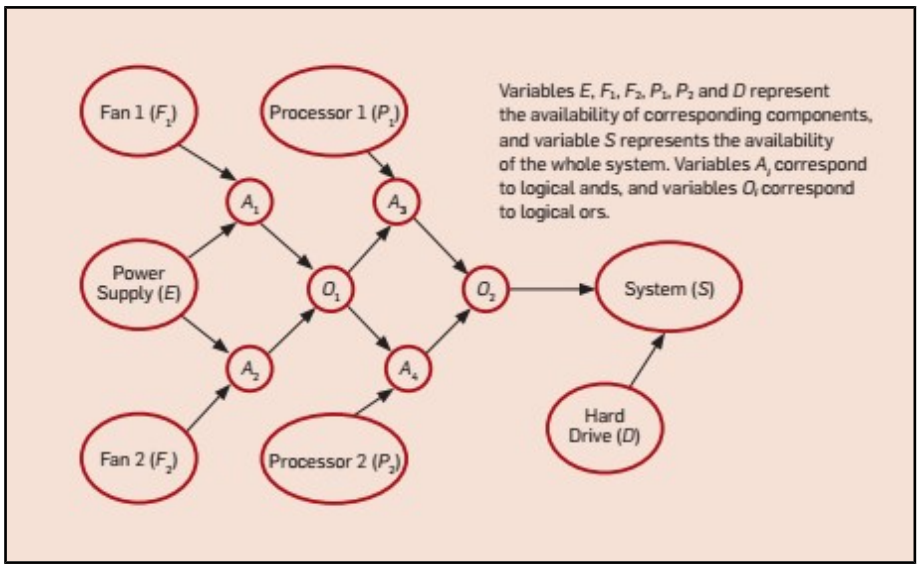


Figure 4-20 A reliability block diagram (top), with a systematic method for mapping its blocks into Bayesian network fragments (bottom).

Consider Figure 4-5 and the data set depicted in the table here as an example.

Sex S	Condition C	Test T_1	Test T_2
male	no	?	–ve
male	?	–ve	+ve
female	yes	+ve	?
⋮	⋮	⋮	⋮

Each row of the table corresponds to an individual and what we know about them. One can use such a data set to learn the network parameters given its structure, or learn both the structure and its parameters. Learning parameters only is an easier task computationally. Moreover, learning either structure or parameters always becomes easier when the data set is complete – that is, the value of each variable is

known in each data record. Since learning is an inductive process, one needs a principle of induction to guide the learning process. The two main principles for this purpose lead to the maximum likelihood and Bayesian approaches to learning. The maximum likelihood approach favors Bayesian networks that maximize the probability of observing the given data set. The Bayesian approach [CITATION ADa03 \l 1040] on the other hand uses the likelihood principle in addition to some prior information which encodes preferences on Bayesian networks [CITATION Adn09 \l 1040]. Suppose we are only learning network parameters.

The Bayesian approach allows one to put a prior distribution on the possible values of each network parameter. This prior distribution, together with the data set, induces a posterior distribution on the values of that parameter. One can then use this posterior to pick a value for that parameter (for example, the distribution mean). Alternatively, one can decide to avoid committing to a fixed parameter value, while computing answers to queries by averaging over all possible parameter values according to their posterior probabilities. It is critical to

observe here that the term “Bayesian network” does not necessarily imply a commitment to the Bayesian approach for learning networks. This term was coined to emphasize three aspects: the often subjective nature of the information used in constructing these networks; the reliance on Bayes conditioning when reasoning with Bayesian networks; and the ability to perform causal as well as evidential reasoning on these networks, which is a distinction underscored by Thomas Bayes. These learning approaches are meant to induce Bayesian networks that are meaningful independently of the tasks for which they are intended. Consider for example a network which models a set of diseases and a corresponding set of symptoms. This network may be used to perform diagnostic tasks, by inferring the most likely disease given a set of observed. It may also be used for prediction tasks, where we infer the most likely symptom given some diseases. If we concern ourselves with only one of these tasks, say diagnostics, we can use a more specialized induction principle that optimizes the diagnostic performance of the learned network. In machine learning jargon, we say we are learning a discriminative model in this case, as it is often used to discriminate among patients according to a predefined set of classes (for example, has cancer or not). This is to be contrasted with learning a generative model, which is to be evaluated based on its ability to generate the given data set, regardless of how it performs on any particular task. We finally note that it is not uncommon to assume some canonical network structure when learning Bayesian networks from data, in order to reduce the problem of learning structure and parameters to the easier problem of learning parameters only. Perhaps the most common such structure is what is known as naïve Bayes: $C \rightarrow A_1, \ldots, C \rightarrow A_n$, where C is known as the class variable and variables $A_1,\ldots, A_n$ are known as attributes. This structure has proven to be very popular and effective in a number of applications, in particular classification and clustering.

4.4 Canonical Bayesian Networks

A number of canonical Bayesian networks have been proposed for modeling some well-known problems in a variety of fields. For example, genetic linkage analysis is concerned with mapping genes onto a chromosome, utilizing the fact that the distance between genes is inversely proportional to the extent to which genes are linked (two genes are linked when it is more likely than not that their states are passed together from a single grandparent, instead of one state from each grandparent). To assess the likelihood of a linkage hypothesis, one uses a pedigree with some information about the genotype and phenotype of associated individuals. Such information can be systematically translated into a Bayesian network [CITATION ADa03 \l 1040] (see Figure 4-6) where the likelihood of a linkage hypothesis corresponds to computing the probability of an event with respect to this network. By casting this problem in terms of inference on Bayesian networks, and by capitalizing on the stateof-the-art algorithms for this purpose, the scalability of genetic linkage analysis was advanced considerably, leading to the most efficient algorithms for exact linkage computations on general pedigrees (for example, the SUPER-LINK program initiated by Fishelson and Geiger). Canonical models also exist for modeling the problem of passing information over a noisy channel, where the goal here is to compute the most likely message sent over such a channel, given the channel output [CITATION Kol09 \l 1040][CITATION Jen01 \l 1040].

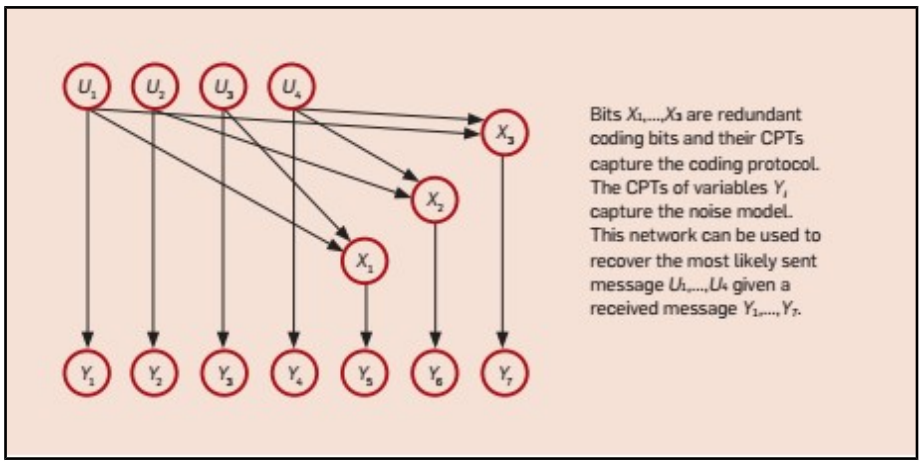


Figure 4-21 A Bayesian Network that models a noisy channel with input (U1, … , U4, X1, …, X3) and output (Y1, …, Y7)

For example, Figure 4-6 depicts a Bayesian network corresponding to a situation where seven bits are sent across a noisy channel (four original bits and three redundant ones). Another class of canonical Bayesian networks has been used in various problems relating to vision, including image restoration and stereo vision.

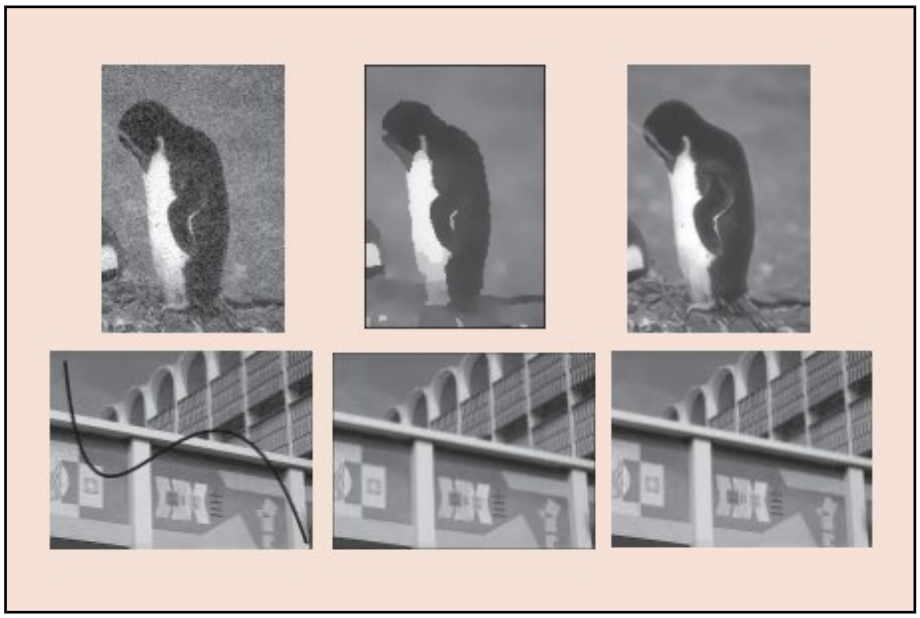


Figure 4-22 Images from left to right: input, restored (using Bayesian Network inference) and original.

Figure 4-7 depicts two examples of images that were restored by posing a query to a corresponding Bayesian network.

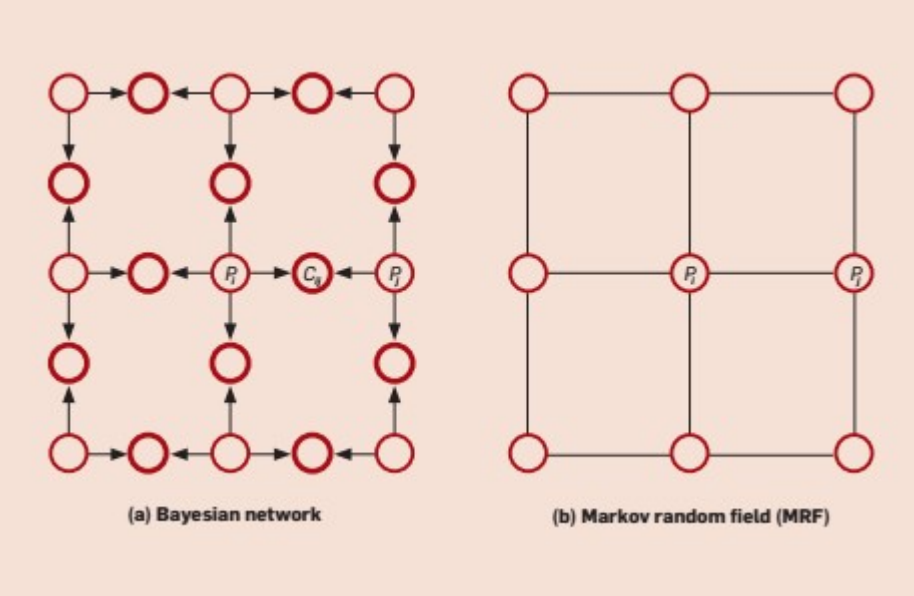


Figure 4-23 Modeling low-level vision problems using two types of graphical models: Bayesian Networks and MRF's.

Figure 4-8a depicts the Bayesian network in this case, where we have one node P_i for each pixel i in the image – the values pi of P_i represent the gray levels of pixel i. For each pair of neighboring pixels, i and j, a child node C_{ij} is added with a CPT that imposes a smoothness constraint between the pixels. That is,

the probability $Pr(C_{ij} = true| P_i = p_i, P_j = p_j)$ specified by the CPT decreases as the difference in gray levels | pi – pj | increases. The only additional information information needed to completely specify the Bayesian network is a CPT for each node P_i, which provides a prior distribution on the gray levels of each pixel i. These CPTs are chosen to give the highest probability to the gray level v_i appearing in the input image, with the prior probability $Pr (P_i = p_i)$ decreasing as the difference $| p_i – v_i |$ increases. By simply adjusting the domain and the prior probabilities of nodes P_i, while asserting an appropriate degree of smoothness using variables C_{ij}, one can use this model to perform other "pixel-labeling" tasks such as stereo vision, photomontage, and binary segmentation. The formulation of these tasks as inference on Bayesian networks is not only elegant, but has also proven to be very powerful. For example, such inference is the basis for almost all top-performing stereo methods.

Canonical Bayesian network models have also been emerging in recent years in the context of other important applications, such as the analysis of documents, and text. Many of these networks are based on topic models that view a document as arising from a set of unknown topics, and provide a framework for reasoning about the connections between words, documents, and underlying topics.

Topic models have been applied to many kinds of documents, including email, scientific abstracts, and newspaper archives, allowing one to utilize inference on Bayesian networks to tackle tasks such as measuring document similarity, discovering emergent topics, and browsing through documents based on their underlying content instead of traditional indexing schemes.

4.5 Bayesian Networks: Use

Similar to any modeling language, the value of Bayesian networks is mainly tied to the class of queries they support. Consider the network in Figure 4-5 for an example and the following queries: Given a male that came out positive on both tests, what is the probability he has the condition? Which group of the population is most likely to test negative on both tests? Considering the network in Figure 4-6: What is the overall reliability of the given system? What is the most likely conFiguretion of the two fans given that the system is unavailable? What single component can be replaced to increase the overall system reliability by 5%? Consider Figure 4-7: What is the most likely channel input that would yield the channel output 1001100? These are example questions that would be of interest in these application domains, and they are questions that can be answered systematically using three canonical Bayesian network queries. A main benefit of using Bayesian networks in these application areas is the ability to capitalize on existing algorithms for these queries, instead of having to invent a corresponding specialized algorithm for each application area [CITATION Adn09 \l 1040].

- **Probability of Evidence**. This query computes the probability Pr(e), where e is an assignment of values to some variables E in the Bayesian network – e is called a variable instantiation or evidence in this case. For example, in Figure 4-6, we can compute the probability that an individual will come out positive on both tests using the probability-of-evidence query Pr(T_1 = +ve, T_2 = +ve). We can also use the same query to compute the overall reliability of the system in Figure 4-6, Pr(S = avail). The decision version of this query is known to be PP-complete. It is also related to another common query, which asks for computing the probability Pr(x|e) for each network variable X and each of its values x. This is known as the node marginals query.

- **Most Probable Explanation (MPE)**. Given an instantiation e of some variables E in the Bayesian network, this query identifies the instantiation q of variables Q = E that maximizes the probability Pr (q| e). In Figure 4-5, we can use an MPE query to find the most likely group, dissected by sex and condition, that will yield negative results for both tests, by taking the evidence e to be T_1 = –ve; T_2 = –ve and Q = {S, C}. We can also use an MPE query to restore images as shown in Figures 4-6 and 4-7. Here, we take the evidence e to be C_{ij} = true for all i, j and Q to include P_i for all i. The decision version of MPE is known to be NP-complete and is therefore easier than the probability-of-evidence query under standard assumptions of complexity theory.

- **Maximum a Posteriori Hypothesis (MAP)**. Given an instantiation e of some variables E in the Bayesian network, this query identifies the instantiation q of some variables Q ⊆ E that maximizes the probability Pr (q|e) [CITATION Mic \l 1040]. Note the subtle difference with MPE queries: Q is a subset of variables E instead of being all of these variables. MAP is a more difficult problem than MPE since its decision version is known to be NPPP -complete, while MPE is only NPcomplete. As an example of this query, consider Figure 4-7 and suppose we are interested in the most likely conFiguretion of the two fans given that the system is unavailable. We can find this conFiguretion using a MAP query with the evidence e being S = un_avail and Q = {F1, F2}.

One can use these canonical queries to implement more sophisticated queries, such as the ones demanded by sensitivity analysis. This is a mode of analysis that allows one to check the robustness of conclusions drawn

from Bayesian networks against perturbations in network parameters. Sensitivity analysis can also be used for automatically revising these parameters in order to satisfy some global constraints that are imposed by experts, or derived from the formal specifications of tasks under consideration. Suppose for example that we compute the overall system reliability using the network in Figure 4-7 and it turns out to be 95%. Suppose we wish this reliability to be no less than 99%: Pr (S = avail) ≥ 99%. Sensitivity analysis can be used to identify components whose reliability is relevant to achieving this objective, together with the new reliabilities they must attain for this purpose. Note that component reliabilities correspond to network parameters in this example.

4.6 How well a Bayesian Network scale?

Algorithms for inference on Bayesian networks fall into two main categories: exact and approximate algorithms. Exact algorithms make no compromises on accuracy and tend to be more expensive computationally. Much emphasis was placed on exact inference in the 1980s and early 1990s, leading to two classes of algorithms based on the concepts of *elimination* and *conditioning [CITATION Dec96 \l 1040][CITATION ADa03 \l 1040]*. In their pure form, the complexity of these algorithms is exponential only in the network *treewidth*, which is a graph-theoretic parameter that measures the resemblance of a graph to a tree structure. For example, the treewidth of trees is ≤ 1 and, hence, inference on such tree networks is quite efficient. As the network becomes more and more connected, its treewidth increases and so does the complexity of inference. For example, the network in Figure 4-6 has a treewidth of n assuming an underlying image with n × n pixels. This is usually too high, even for modest-size images, to make these networks accessible to treewidthbased algorithms.

The pure form of elimination and conditioning algorithms are called *structure-based* as their complexity is sensitive only to the network structure. In particular, these algorithms will consume the same computational resources when applied to two networks that share the same structure, regardless of what probabilities are used to annotate them. It has long been observed that inference algorithms can be made more efficient if they also exploit the structure exhibited by network parameters, including determinism (0 and 1 parameters) and contextspecific independence (independence that follows from network parameters and is not detectable by d-separation). Yet, algorithms for exploiting parametric structure have only matured in the last few years, allowing one to perform exact inference on some networks whose treewidth is quite large.

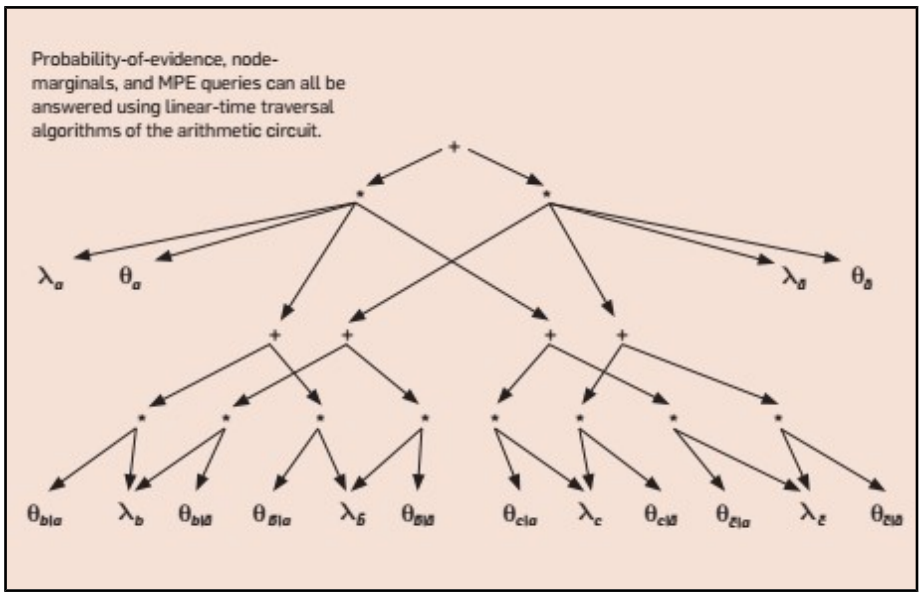


Figure 4-24 An arithmetic circuit for the Bayesian network B <- A -> C. Inputs labeled with θ variables correspond to network parameters, while those labeled with λ variables capture evidence.

Networks that correspond to genetic linkage analysis (Figure 4-9) tend to fall in this category and so do networks that are synthesized from relational models. One of the key techniques for exploiting parametric structure is based on compiling Bayesian networks into arithmetic circuits, allowing one to reduce probabilistic inference to a process of circuit propagation; see Figure 4-8 (see the previous page). The size of these compiled circuits is determined by both the network topology and its parameters, leading to relatively compact circuits in some situations where the parametric structure is excessive, even if the network treewidth is quite high. Reducing inference to circuit propagation makes it also easier to support applications that require real-time inference, as in certain diagnosis applications. Around the mid-1990s, a strong belief started forming in the inference community that the performance of exact algorithms must be exponential in treewidth – this is before parametric structure was being exploited effectively. At about the same time, methods for automatically constructing Bayesian networks started maturing to the point of yielding networks whose treewidth is too large to be handled efficiently by exact algorithms at the time. This has led to a surge of interest in approximate inference algorithms, which are generally independent of treewidth. Today, approximate inference algorithms are the only choice for networks that have a high treewidth, yet lack sufficient parametric structure – the networks used in low-level vision applications tend to have this property. An influential class of approximate inference algorithms is based on reducing the inference problem to a constrained optimization problem, with loopy belief propagation and its generalizations as one key example. Loopy belief propagation is actually the common algorithm of choice today for handling networks with very high treewidth, such as the ones arising in vision or channel coding applications. Algorithms based on stochastic sampling have also been pursued for a long time and are especially important for inference in Bayesian networks that contain continuous variables. Variational methods provide another important class of approximation techniques and are key for inference on some Bayesian networks, such as the ones arising in topic models.

4.7 Casuality

One of the most intriguing aspects of Bayesian networks is the role they play in formalizing causality. To illustrate this point, consider Figure 4-10, which depicts two Bayesian network structures over the same set of variables.

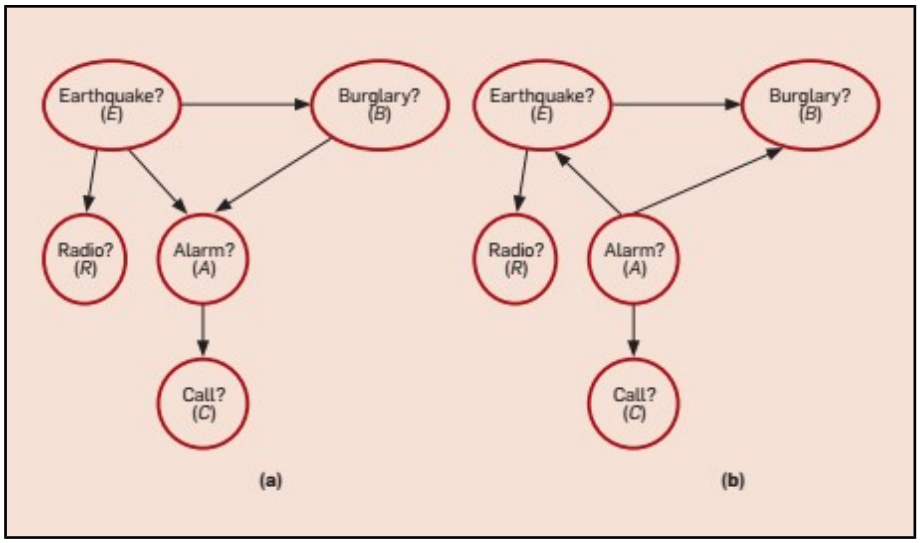


Figure 4-25 Two networks that represent the same set of conditional independencies.

One can verify using d-separation that these structures represent the same set of conditional independencies. As such, they are representationally equivalent as they can induce the same set of probability distributions when augmented with appropriate CPTs. Note, however, that the network in Figure 4-10a is consistent with common perceptions of causal influences, yet the one in Figure 4-10b violates these perceptions due to edges A → E and A → B. Is there any significance to this discrepancy? In other words, is there some additional information that can be extracted from one of these networks, which cannot be extracted from the other? The answer is yes according to a body of work on causal Bayesian networks, which is concerned with a key question: how can one characterize the additional information captured by a causal Bayesian network and, hence, what queries can be answered only by Bayesian networks that have a causal interpretation? According to this body of work, only causal networks are capable of updating probabilities based on interventions, as opposed to observations.

To give an example of this difference, consider Figure 4-9 again and suppose that we want to compute the probabilities of various events given that someone has tampered with the alarm, causing it to go off. This is an intervention, to be contrasted with an observation, where we know the alarm went off but without knowing the reason. In a causal network, interventions are handled as shown in Figure 4-10a: by simply adding a new direct cause for the alarm variable. This local fix, however, cannot be applied to the non-causal network in Figure 4-10b. If we do, we obtain the network in Figure 4-10b, which asserts the following (using d-separation): if we observe the alarm did go off, then knowing it was not tampered with is irrelevant to whether a burglary or an

earthquake took place. This independence, which is counterintuitive, does not hold in the causal structure and represents one example of what may go wrong when using a non-causal structure to answer questions about interventions. Causal structures can also be used to answer more sophisticated queries, such as counterfactuals. For example, the probability of "the patient would have been alive had he not taken the drug" requires reasoning about interventions (and sometimes might even require functional information, beyond standard causal Bayesian networks). Other types of queries include ones for distinguishing between direct and indirect causes and for determining the sufficiency and necessity of causation. Learning causal Bayesian networks has also been treated although not as extensively as the learning of general Bayesian networks.

4.8 Beyond Bayesian Networks

Viewed as graphical representations of probability distributions, Bayesian networks are only one of several other models for this purpose. In fact, in areas such as statistics (and now also in AI), Bayesian networks are studied under the broader class of probabilistic graphical models, which include other instances such as Markov networks and chain graphs. Markov networks correspond to undirected graphs, and chain graphs have both directed and undirected edges. Both of these models can be interpreted as compact specifications ofprobability distributions, yet their semantics tend to be less transparent than Bayesian networks.

For example, both of these models include numeric annotations, yet one cannot interpret these numbers directly as probabilities even though the whole model can be interpreted as a probability distribution. Figure 4-6b depicts a special case of a Markov network, known as a Markov random field (MRF), which is typically used in vision applications. Comparing this model to the Bayesian network in Figure 4-7a, one finds that smoothness constraints between two adjacent pixels P_i and P_j can now be represented by a single undirected edge $P_i - P_j$ instead of two directed edges and an additional node, $P_i \rightarrow C_{ij} \leftarrow P_j$. In this model, each edge is associated with a function f (P_i, P_j) over the states of adjacent pixels. The values of this function can be used to capture the smoothness constraint for these pixels, yet do not admit a direct probabilistic interpretation. Bayesian networks are meant to model probabilistic beliefs, yet the interest in such beliefs is typically motivated by the need to make rational decisions. Since such decisions are often contemplated in the presence of uncertainty, one needs to know the likelihood and utilities associated with various decision outcomes. A classical example in this regard concerns an oil wildcatter that needs to decide whether or not to drill for oil at a specific site, with an additional decision on whether to request seismic soundings that may help determine the geological structure of the site. Each of these decisions has an associated cost. Moreover, their potential outcomes have associated utilities and probabilities.

The need to integrate these probabilistic beliefs, utilities and decisions has lead to the development of Influence Diagrams, which are extensions of Bayesian networks that include three types of nodes: chance, utility, and decision. Influence diagrams, also called decision networks, come with a toolbox that allows one to compute optimal strategies: ones that are guaranteed to produce the highest expected utility. Bayesian networks have also been extended in ways that are meant to facilitate their construction. In many domains, such networks tend to exhibit regular and repetitive structures, with the regularities manifesting in both CPTs and network structure. In

these situations, one can synthesize large Bayesian networks automatically from compact high-level specifications. A number of concrete specifications have been proposed for this purpose. For example, template-based approaches require two components for specifying a Bayesian network: a set of network templates whose instantiation leads to network segments, and a specification of which segments to generate and how to connect them together. Other approaches include languages based on first-order logic, allowing one to reason about situations with varying sets of objects.

4.9 The challenges ahead

Bayesian networks have been established as a ubiquitous tool for modeling and reasoning under uncertainty. The reach of Bayesian networks, however, is tied to their effectiveness in representing the phenomena of interest, and the scalability of their inference algorithms. To further improve the scope and ubiquity of Bayesian networks, one therefore needs sustained progress on both fronts. The main challenges on the first front lie in increasing the expressive power of Bayesian network representations, while maintaining the key features that have proven necessary for their success: modularity of representation, transparent graphical nature, and efficiency of inference. On the algorithmic side, there is a need to better understand the theoretical and practical limits of exact inference algorithms based on the two dimensions that characterize Bayesian networks: their topology and parametric structure.

With regard to approximate inference algorithms, the main challenges seem to be in better understanding their behavior to the point where we can characterize conditions under which they are expected to yield good approximations, and provide tools for practically trading off approximation quality with computational resources. Pushing the limits of inference algorithms will immediately push the envelope with regard to learning Bayesian networks since many learning algorithms rely heavily on inference. One cannot emphasize enough the importance of this line of work, given the extent to which data is available today, and the abundance of applications that require the learning of networks from such data.

5 SOFTWARE FOR GRAPHICAL MODELS

Graphical models (GMs) are a way to represent conditional independence assumptions by using graphs. Specifically, nodes represent random variables and lack of edges represent conditional independencies. The graph is a useful visual representation of complex stochastic systems. The graphical structure is also the basis of efficient inference algorithms. There are many different kinds of graphical models, but the two most popular ones are based on directed acylic graphs (also called "Bayesian networks") and on undirected graphs (also called "Markov random fields"). In this article, we review some of the more popular and/or recent software packages for dealing with graphical models. A more extensive comparison can be found in the last section of this chapter. In this section we review some of the major software packages for Bayesian and decision network modeling and inference. The information should be read in conjunction with the last section of this chapter. The additional aspects we considerare as follows.

- **Development**: Any background information of the developers or the history of this software.
- **Technical**: Further information about platforms or products (beyond the summary given in the final table).
- **Node Types**: Relating to discrete/continuous support.
- **CPTs**: Support for elicitation, or local structure.
- **Inference**: More details about the inference algorithm(s) provided, and possible user control over inference options. Also whether computes MPE and P(E).
- **Evidence**: Whether negative and likelihood evidence are supported, in addition to specific evidence.
- **Decision networks**: Information about decision network evaluation (if known), whether expected utilities for all policies are provided, or just decision tables. Whether precedence links between decision nodes are determined automatically if not specified by the knowledge engineer. Also, whether value of information is supported directly.
- **DBNs**: Whether DBN representation and/or inference is supported.
- **Learning**: What learning algorithms are used.
- **Evaluation**: What support, if any, for evaluation?. In particular, sensitivity analysis and statistical validation methods.
- **Other features**: Functionality not found in most other packages.

5.1 BUGS

BUGS (Bayesian inference using Gibbs Sampling) assumes the model is specified in the form of a DAG (directed acyclic graph), and uses Gibbs sampling for inference. A large number of different conditional distributions (node types) are supported. Internally, various algorithms (such as adaptive rejection sampling and slice sampling) are used to sample from the full conditionals. The software is easy to use, especially since it has recently become possible to call it directly from R (using R2WinBUGS2) and Matlab (using Mat-BUGS3), thus bypassing the rather cumbersome GUI. Unfortunately, single site Gibbs sampling can be very slow to "mix", resulting in unreliable posterior inferences. In addition, Gibbs sampling cannot be used to find posterior modes, and cannot easily be used to compute the marginal likelihood, which is useful for model selection (although BUGS does return the DIC score of a model). BUGS is freely available as an executable file. The most recent version, called WinBUGS, only runs on Windows (although one can run it on Linux systems using Wine). Recently, an opensource

alternative called OpenBUGS4 has been created, but it is not nearly as mature as Win-BUGS. OpenBUGS is written in a language called "Component Pascal".

5.2 JAGS

JAGS (Just Another Gibbs Sampler)5 is very similar in functionality to BUGS. The main difference is that it is fully open source, and works easily on multiple platforms (Windows, unix, etc). The principle advantage over OpenBUGS is that it is written in Java, which is a more widely known language than Component Pascal. In addition, it seems to have a simpler design than OpenBUGS.

5.3 VIBES

VIBES (Variational Inference for Bayesian Networks) is open-source Java, and is designed to be similar to BUGS in functionality, but it uses the variational mean field algorithm for inference. This is potentially much faster, but less accurate than Gibbs sampling. In addition, it is limited to the conjugate exponential family. Note that VIBES is no longer being supported; its author is developing a replacement called Infer.NET (see below)

5.4 Infer.NET

Infer.NET is a software package developed at Microsoft Research in Cambridge. They anticipate an initial public release in Spring 2008. The code will not be open source but will be freely available for academic use. The

model is specified using a new programming language called *Csoft*, which allows one to combine stochastic code with standard C# code. Thus one can easily specify graphical models of various kinds. Various Bayesian inference algorithms are supported, including Gibbs sampling, variational mean field, and expectation propagation. The package is designed to generate model-specific code, and to run very fast, even on large models.

5.5 BNT

BNT (Bayes Net Toolbox) is open-source Matlab, and supports many different models and inference algorithms. In particular, it supports DAG models, "dynamic Bayesian networks" (which are DAG models unrolled in time) and influence/decision diagrams. It also has undocumented and partial support for undirected models. In terms of inference, BNT, like many other GM packages, can only perform Bayesian inference on discrete or Gaussian random variables. Hence parameter inference is performed using point estimation techniques such as EM or gradient descent. However, conditional on the parameters, inference of the remaining variables can often be performed exactly, using the junction tree algorithm, which includes well-known algorithms, such as the forwards-backwards algorithm, as special cases. If exact methods are too slow, a variety of different approximate inference algorithms are supported, such as "loopy belief propagation".

- **Development**: This package was developed during Kevin Murphy's time at U.C. Berkeley as a Ph.D. student, where his thesis addressed DBN representation, inference and learning. He also worked on BNT while an intern at Intel.
- **Technical**: The Bayes Net Toolbox is for use with only with Matlab, a widely used and powerful mathematical software package. Its lack of a GUI is made up for by Matlab's visualization features. This software is distributed under the Gnu Library General Public License.
- **CPTs**: BNT support the following conditional probability distributions: tabular (multinomial), Gaussian, Softmax (logistic/ sigmoid), Multi-layer perceptron (neural network), Noisy-or, Deterministic.
- **Inference**: BNT supports many different exact and approximate inference algorithms, for both ordinary BNs and DBNs, including all the algorithms described in this text.
- **DBNs**: The following dynamic models can be implemented in BNT: Dynamic HMMs, Factorial HMMs, coupled HMMs, input-output HMMs, DBNs, Kalman filters, ARMAX models, switching Kalman filters, tree-structured Kalman filters, multiscale AR models.
- **Learning**: BNT parameter learning methods are: Batch MLE/MAP parameter learning using EM (different M and E methods for each node type); Sequential/ batch Bayesian parameter learning (for tabular nodes only). Structure learning methods are: Bayesian structure learning, using MCMC or local search (for fully observed tabular nodes only); Constraint-based structure learning (IC/PC and IC*/FCI).

5.6 Hugin

Hugin is a commercial package with functionality similar to BNT. It was one of the first packages for DAG models (including influence diagrams), and it is arguably the most mature. However, there are now a large number of other packages, such as Genie, MSBNx, Netica, PNL, etc. with very similar functionality. These packages focus on exact inference in discrete-state (or conditionally Gaussian) models, using the junction tree or variable elimination algorithm. Some of them also support parameter estimation using EM. These packages are aimed at the business/ data-mining market, and hence they often put more emphasis on user interface and I/O issues than on core functionality.

- **Development**: The original Hugin shell was initially developed by a group at the Aalborg University, as part of an ESPRIT project which also producedMUNIN system. Hugin's development continued through another Lauritzen-Jensen project called ODIN. Hugin Expert was established to start commericializing the Hugin tool. The close connection between Hugin Expert and the Aalborg research group has continued, including co-location and personnel moving between the two. This has meant that Hugin Expert has consistently contributed to and taken advantage of the latest BN research. In 1998 Hewlett-Packard purchased 45% of Hugin Expert; one consequence of this seems to have been the tailored development of Hugin to support trouble-shooting.
- **Technical**: The Hugin API is called the "Hugin Decision Engine". It is available for the languages C++, Java and as an ActiveX-server, and runs on the operating systems: Sun Solaris (Sparc and x86), HP-UX, Linux, and Windows. Versions are available for single and double-precision floating-point operations. The Hugin GUI (called "Hugin") is available for Sun Solaris (sparc, x86)Windows, and Linux red-hat. Hugin also offers "Hugin Advisor" for developing trouble shooting applications, and "Hugin Clementine" for integrating Hugin's learning with datamining in SPSS's Clementine system.
- **Node Types**: Good support for continuous variable modeling, and combining discrete and continuous nodes, following on from research in this area. CPTs: CPTs can be specified with expressions as well as through manual entry. The CPTs don't have to sum to one; entries that don't sum to one are normalized.
- **Inference**: The basic algorithm is the junction tree algorithm, with options to choose between variations. The junction tree may be viewed. There is the option to vary the triangulation method, and another to turn on compression (of zeros in the junction tree). In addition Hugin GUI computes P(E), the data conflict measure and the MPE.
- **Evidence**: Specific, negative and virtual evidence are all supported. Decision networks: Hugin requires the existence of a directed path including all decision variables. It gives the expected utility of each decision option in the decision table.
- **Learning**: The parameter learning is done with EM learning and Spiegelhalter & Lauritzen sequential learning (adaptation) and fading are also supported. Structure learning is done using the PC algorithm.
- **Other features**: Supports object-oriented BNs.

5.7 gR

gR (graphical models in R) is a collection of packages rather than a single package. The main package is gRbase, which is a way of defining data and models. There is also the dynamicGraph package, for visualizing and editing graphs. There are no Bayesian inference alorithms implemented in R. However, R interfaces to several existing model-fitting packages are provided, including CoCo, and mimR, both of which are designed for fitting contingency tables, which can be represented as undirected GMs.

5.8 Blaise

Blaise is a Java software package that supports efficient Monte Carlo inference (including MCMC and sequential Monte Carlo samplers) in a large class of probabilistic models, including directed graphical models and non-parametric Bayesian models. The plan is to release a first version to the public, under a restricted open source license, in Spring 2008.

5.9 Gaussian Graphical Models

Gaussian graphical models are an important special case of graphical models that support efficient Bayesian inference using techniques from sparse linear algebra. GMRFsim13 supports inference in undirected GGMs, and GDAGSim supports inference in directed GGMs. Both of these can be used to perform block sampling inside an MCMC sampler. The ggm R package can be used to fit undirected GGMs parameters using point estimation techniques.

5.10 Model Selection

In addition to inference about states and parameters, there is much interest (especially in the systems biology community) in inference about the graph structure itself. The model selection problem is very difficult, because the space of all graphs on n nodes has size $O(2^{n^2})$. There are basically three main approaches to this: greedy search (and variants), MCMC model averaging, and constraint-based methods. Most of the work has focused on learning DAG models, although the *WinMine* package learns dependency networks. There are many packages that perform greedy search in DAG space: *BNT*, *DAGlearn*, *Banjo*, *Deal*, etc. *BNT* supports simple hillclimbing. DAGlearn uses L1-penalized logistic regression to reduce the search space. Banjo uses simulated annealing. Deal uses hill-climbing, but can handle conditionally Gaussian models (the other packages assume discrete data). There are very few publically available packages that perform Bayesian model averaging in the space of DAGs. *BNT* implements a Metropolis Hastings method with a simple local proposal. *BDAGL* uses a more sophisticated proposal based on dynamic programming. The GGM package does model averaging in the space of

undirected Gaussian GMs, using MCMC and stochastic search techniques. The *HdBCS* (high dimensional Bayesian covariance selection) package is similar to the GGM package, but searches in the space of DAGs and then converts the result to an undirected GGM. The constraint-based approach to structure learning, in which one eliminates edges if certain conditional independencies are detected in the data (using some hypothesis testing procedure), is generally faster but more error-prone than the above Bayesian techniques. *BNT* implements some of the simpler algorithms. *Tetrad* is a more elaborate package. Traditionally, the constraint-based approach has been the method of choice for people interested in learning "causal" models from observational data. However, this approach can of course be used to fit "acausal" models, too. For example, the *SIN* R package uses conditional independency tests to learn the structure of undirected Gaussian GMs. The *GeneNet* R uses an FDR approach to threshold the partial correlation coefficients to induce a sparse GGM. The *glasso* R package uses L1 regularization to estimate a sparse precision matrix.

5.11 WinMine

WinMine is a set of causal discovery programs for Windows 2000/NT/XP. The majority of the programs are command-line executables that can be run in scripts. It includes GUIs for viewing learned or modeled Bayesian networks and for displaying classification trees ("decision trees"). It is freely downloadable, if it is not going to be used for commercial purposes. WinMine can learn discrete Bayesian networks from sample data. It supports prior information in the form of partial variable orderings and forbidden arcs. There is also support for evaluating the learned models.

5.12 TETRAD

TETRAD II was the first commercially available causal discovery program. It is available for purchase, with relevant information available on the website. TETRAD III adds Gibbs sampling to the functionality of TETRAD II, but retains the command-line interface. TETRAD IV has a graphical interface, running under the Java Runtime Environment. TETRAD III and IV are available for free download.

5.13 CaMML

CaMML (Causal discovery via MML) is freely available as an executable download for Linux from the web site. There are two different versions: CaMML-L, which learns linear Gaussian models, and CaMML, which learns discrete causal models. Both use the Metropolis sampling search algorithm. These versions have a fairly crude ASCII command-line interface. There is a project to reimplement CaMML inside the Monash CDMS (Core Data Mining Software) project, providing CaMML with a GUI interface and data visualization capabilities. When ready, this will be available from the web site, as will any future developments from the CaMML project.

5.14 Graphical Models Software Packages Comparison Table

Here explained the meaning of the headers in the table:

- **Src**. Source code included? (N=no) If so, what language?
- **API**. Application program interface included? (N means the program cannot be integrated into your code, i.e., it must be run as a standalone executable.)
- **Exec**. Executable runs on W = Windows (95/98/NT), U = Unix, M = Mac, or - = any machine with a compiler.
- **Cts**. Are continuous (latent) nodes supported? G = (conditionally) Gaussians nodes supported analytically, Cs = continuous nodes supported by sampling, Cd = continuous nodes supported by discretization, Cx = continuous nodes supported by some unspecified method, D = only discrete nodes supported.
- **GUI**. Graphical User Interface included?
- **Par.** Learns parameters?
- **Str.** Learns structure? CI = means uses conditional independency tests
- **Utl**. Utility and decision nodes (i.e., influence diagrams) supported?
- **$.** 0 = free (although possibly only for academic use). $ = commercial software (although most have free versions which are restricted in various ways, e.g., the model size is limited, or models cannot be saved, or there is no API.)
- **Uni.** What kind of graphs are supported? U = only undirected graphs, D = only directed graphs, UD = both undirected and directed, CG = chain graphs (mixed directed/undirected).
- **Inf**. Which inference algorithm is used? jtree = junction tree, varelim = variable (bucket) elimination, MH = Metropols Hastings, G = Gibbs sampling, IS = importance sampling, sampling = some other Monte Carlo method, polytree = Pearl's algorithm restricted to a graph with no cycles, VMP = variational message passing, EP = expectation propagation, SL = the program is designed for structure learning from completely observed data, not state estimation
- **Comments**. If in "quotes", I am quoting the authors at their request.

Name	Src	API	Exec	Cts	GUI	Par	Str	Utl	$	Un	Inf	Comments
AgenaRisk	N	Y	W,U	Cx	Y	Y	N	N	$	D	JTree	Simulation by Dynamic discretisation
Analytica	N	Y	W,M	G	Y	N	N	Y	$	D	sampling	spread sheet compatible

B-course	N	N	W,U, M	Cd	Y	Y	Y	N	0	D	?	Runs on their server: view results using a web browser.
Banjo	Java	Y	W,U, M	Cd	N	N	Y	N	0	D	none	structure learning of static or dynamic networks of discrete variables
Bassist	C++	Y	U	G	N	Y	N	N	0	D	MH	Generates C++ for MCMC. (No longer maintained)
BayesBuilder	N	N	W	D	Y	N	N	N	0	D	?	-
BayesiaLab	N	N	-	Cd	Y	Y	Y	N	$	CG	jtree,G	Structural learning, adaptive questionnaires, dynamic models
Bayes-Scala	Scala	Y	-	D	N	Y	N	N	Y	UD	Loopy BP	"Loopy BP in Cluster Graph, EM learning (BN, unrolled DBN, incomplete data)"
Bayes Server	N	Y	W	G	Y	Y	N	N	$	D	RelevanceTr ee,Varelim	Supports inference and learning with Dynamic Bayesian networks and continuous variables
Bayesware Discoverer	N	N	WUM	Cd	Y	Y	Y	N	$	D	?	Uses bound and collapse for learning with missing data.
BayesBlocks	Python /C++	Y	-	Y	N	Y	N	N	0	Dir	Variational	Non-Gaussian Latent variable models
Blaise	Java	Y	-	Y	N	Y	N	N	0	Fgr aph	MCMC, SMC	General MC toolkit, also handles non-parametric Bayesian models
BNT	Matlab /C	Y	WUM	G	N	Y	Y	Y	0	D,U	Many	Also handles dynamic models, like HMMs and Kalman filters.
BNJ	Java	-	-	D	Y	N	Y	N	0	D	jtree, IS	-

BNL	Matlab	-	-	D	N	N	N	N	0	D	jtree	Supports (ordinal) logistic regression CPDs and EM learning
BUGS	N	N	WU	Cs	W	Y	N	N	0	D	Gibbs	-
CoCo+Xlisp	C/lisp	Y	U	D	Y	Y	CI	N	0	U	Jtree	Designed for contingency tables.
CIspace	Java	N	WU	D	Y	N	N	N	0	D	Varelim	-
CRFtoolbox	Matlab /C	Y	-	N	N	Y	N	N	0	U	Loopy BP	Conditional random fields, arbitrary structure
DBNbox	Matlab	-	-	Y	N	Y	N	N	Y	D	Various	DBNs
Deal	R	-	-	G	Y	Y	Y	N	0	D	None	Structure learning.
DeriveIt	N	-	-	?	?	Y	Y	?	$	D	Jtree, Gibbs	Exploits local structure in CPDs.
Dimple	MATLAB/Java	MATLAB and Java	W,U,M	G, Cs, Cd	N	Y	N	N	0	UD, CG	BP,G,MH	Parameter learning is mostly limited to EM for now
Elvira	Java	Y	W,U,M	Cd, Cx	Y	Y	Y	Y	0	D	JTree,varelim,IS	"Also includes classification, abductive inference and model fusion"
Ergo	N	Y	W,M	D	Y	N	N	N	$	D	jtree	-
FastInf	C++	Y	U,M	D	N	Y	N	N	0	U	JTree,G,VMP	Also supports GBP,TRBP
Figaro	Scala	N	-	-	-	Y	Y	Y	$	U	S	Also supports first order models
GDAGsim	C	Y	WUM	G	N	N	N	N	0	D	Exact	Bayesian analysis of large linear Gaussian directed models.

GeNIe and SMILE	SMILE wrappers only	Y	W,U,M,other	Cs,equations	W,U,M	Y	Y	Y	0	D	JTree,sampling	DBNs, support for diagnostic applications
GGM	C++	-	-	G	N	Y	Y	N	0	U	SL	MCMC and stochastic search for structure learning of GGMs
GMRFsim	C	Y	WUM	G	N	N	N	N	0	U	MCMC	Bayesian analysis of large linear Gaussian undirected models.
GMTk	N	Y	U	D	N	Y	Y	N	0	D	Jtree	Designed for speech recognition.
GOBNILP	Y	N	-	D	N	N	Y	N	0	D	none	"exact structure learning from data or local scores, k-best learning possible"
gR	R	-	-	-	-	-	-	-	0	-	-	Various packages
Grappa	R	-	-	D	N	N)	N	N	0	D	Jtree	-
HdBCS	C++	-	-	G	N	Y	Y	N	0	U	SL	stochastic search for structure learning of GGMs
Hugin Expert	N	Y	W	G	W	Y	CI	Y	$	CG	Jtree	-
Hydra	Java	-	-	Cs	Y	Y	N	N	0	U,D	MCMC	-
IBAyes	N	N	W	D	Y	N	N	N	0	D	Junction Tree, Sampling	-
Infer.NET	C#	Y	Y	Y	N	Y	N	N	0	Y	VMP, EP, Gibbs	Bayesian parameter estimation as well
JAGS	Java	Y	-	Y	N	Y	N	N	0	Y	Gibbs	Similar to BUGS
Java Bayes	Java	Y	WUM	D	Y	N	N	Y	0	D	Varelim, jtree	-
LADR	N	Y	-	Cd	N	N	Y	N	$	D	none	"Structure learning of massive static or dynamic networks"

LibB	N	Y	W	D	N	Y	Y	N	0	D	SL	Structure learning
libDAI	C++	Y	-	D	N	Y	N	N	0	Fgraph	JTree,VarElim,G,VMP	also supports GBP,HAK,LCBP,TreeEP,TRWBP
Libra	OCaml	N	-	D	N	Y	Y	N	0	UD	varelim, Gibbs, loopy BP, VMP	Also supports dependency networks, arithmetic circuits, and sum-product networks.
MIM	N	N	W	G	Y	Y	Y	N	$	CG	Jtree	Up to 52 variables.
Mocapy++	C++	Y	W,U,M	G	N	Y	N	N	0	D	Gibbs sampling	Support for directional statistics
MSBNx	N	Y	W	D	W	N	N	Y	0	D	Jtree	-
Netica	N	WUM	W	G	W	Y	N	Y	$	D	jtree	-
OpenGM2	C++, Matlab, Python	Y	WUM	D	Y	N	N	N	0	Fgraph	Many	LP, ILP, Multicut, LBP, TRBP, QPBO, AStar, many graphcut-based methods, methods based on dual decomposition, and many more. Includes wrappers for several other related projects.
OpenMarkov	Y	Y	Java (U,W,M)	Cs, Cd	Y	Y	Y	Y	Y	UD	jtree,varelim,sampling	"Java, open source, extensible; dynamic models, object oriented networks, interactive learning, ProbModelXML format"
PMT	Matlab/C	-	-	D	N	Y	N	N	0	D	special purpose	-
PNL	C++	-	-	D	N	Y	Y	N	0	U,D	Jtree	A C++ version of BNT; will be released 12/03
Pulcinella	Lisp	Y	WUM	D	Y	N	N	N	0	D	?	Uses valuation systems for non-probabilistic calculi.
RISO	Java	Y	WUM	G	Y	N	N	N	0	D	Polytree	Distributed implementation.

Sam Iam	N	N ?	WU ? (Java executable)	G ?	Y	Y	N ?	Y	0	D	Recursive conditioning	Also does sensitivity Analysis
Stan	C++	C++, R	-	Y	N	Y	N	N	0	D	Hybrid Monte Carlo	Generates efficient MCMC code for BUGS-like models
Tetrad	N	N	WU	G	N	Y	CI	N	0	U,D	SL	-
UC Irvine	Y	N	W,U	D	N	N	N	N	0	UD	AND/OR Search	Bucket Elimination, AND/OR search for P(evidence), MPE in Bayesian networks
UnBBayes	Java	Y	-	G, Cs, Cd	Y	Y	Y	Y	0	D	jtree, G, sampling, VMP	"Supports other probabilistic graphical models: MSBN, OOBN, PRM, and MEBN."
Uninet	N	Y	W	G, Cs, Cx	Y	Y	Y	N	0	UD	sampling	"All probabilsitic nodes (discrete, Gaussian, non-Gaussian) are supported analytically through the Vine-Copula method with Gaussian copula. Functional nodes are supported by sampling."
Vibes	Java	Y	WU	Cx	Y	Y	N	N	0	D	VMP	
WinMine	N	N	W	Cx	Y	Y	Y	N	0	U,D	SL	Learns BN or dependency net structure.
XBAIES 2.0	N	N	W	G	Y	Y	N	Y	0	CG	Jtree	-

6 INTERACTIVE LEARNING OF BAYESIAN NETWORKS USING OPENMARKOV

Algorithms for learning Bayesian networks (BNs) behave as a black box that takes a database as an input and returns a network as the output. In contrast, OpenMarkov, the tool we used to solve our case study (see the next chapter), includes the option to run the algorithms in a step-by-step fashion, presenting a ranked list of operations (such as adding, removing, or inverting links) the user can select, while allowing live edition of the BN throughout the learning process. The application offers some data preprocessing options and the possibility to use a model network to guide the learning process. This functionality in OpenMarkov can be employed to learn BNs with partial expert knowledge, to debug new algorithms, and as a pedagogical tool.

6.1 Introduction

A probabilistic graphical model (PGM) consists of a joint probability distribution defined on a set of variables V and a graph containing a node for each variable X in V; the structure of the graph imposes some relations of conditional independence on the structure of the network, which depend mainly on the type of graph. Some types of PGMs are Bayesian networks (BNs), Markov networks, influence diagrams, hidden Markov models, factored MDPs and POMDPs, etc. In many cases PGMs are built from expert knowledge: causal relations are used to draw the arcs of the graph, and the conditional probabilities are obtained from the literature (for example, medical journals), from databases, or from experts' estimations.

The difficulty and tediousness of this approach has led to an increasing interest for learning methods that can generate PGMs from databases automatically. This is the preferred approach when there is a large database with few or none missing values, and no causal knowledge. However, in many cases the size of the database does not allow to learn a PGM that accurately represents the conditional independencies existing in the domain of application. Practitioners of these methods often encounter that the PGM obtained contains some links that the expert considers as obviously spurious, but given that in general the algorithms perform as black boxes, it is diffcult to determine to what degree those links are really supported by the data.

Another problem is that most learning algorithms do not return causal models. In the last decade the number of studies aimed at obtaining causal models from databases has grown exponentially (the UAI Conference held in

Barcelona in July 2011 was a clear illustration of this phenomenon). However in many cases the problem does not lie in the algorithm but on the lack of information in the database: the

only conclusion that can be drawn reliably from a set of data – provided that it is big enough and not biased – is the set of correlations that exist in the real world. These correlations rule out some causal models, but the number of models compatible with the data is usually very large. Many of those models clash with common knowledge of the experts, but it is not easy to feed that knowledge into automatic causal learning algorithms. For this reason, it would be useful to have interactive learning algorithms that propose a list of changes to improve the accuracy of the network but allow an expert to select only those that do not contradict his/her knowledge. Secondly, interactive learning could be also of great interest for researchers and developers of new algorithms. An interactive learning tool able to show on a graphical interface the different actions that the algorithm is considering at each step and the scores assigned to them may be very useful to debug the algorithm, for example, by observing how a shift in some of the parameters leads to a different selection of actions.

Thirdly, interactive learning programs may have a high pedagogical value by allowing the students to know the actions that the algorithm has evaluated at each step and why it has selected each action. Then it is possible to run a different algorithm, for example with a different search strategy or a different metric, and observe why it selects different actions at each step. For these reasons it's fundamental the implementation of an interactive learning module in OpenMarkov, an open-source software tool for editing and evaluating PGMs. The interactive learning module includes a user-friendly graphical interface that allows to overcome all the above-pointed problems, with the corresponding benefit for experts, researchers, developers and students.

The rest of this chapter is structured as follows. In Section 6.2 we give a brief overview of Bayesian network learning and of the Open-Markov tool. Section 6.3 describes the different options available to the user for learning BNs interactively in OpenMarkov. Section 6.4 presents a case study: how to learn interactively the BN Alarm, a model frequently used in the literature as a benchmark for learning algorithms. In Section 6.5 we discuss the advantages of this and similiar approaches to the field, and Section 6.6 contains the conclusions.

6.2 Learning Bayesian Networks

Learning Bayesian networks is one of the most important research areas in the field of BNs. Every year, around one third of the total publications in that area are related to automatic learning. Just like in manual construction, automatic learning of BNs presents two aspects: parametric learning and structural learning. Parametric learning consists of computing the conditional probabilities given by the structure

of the network using the observed frequencies on the database. Structural learning tries to find the graph that best represents the probability distribution based on the frequencies in the database.

Structural learning methods. There are two main methods for building the graph of a BN from a database. The first one consist of detecting the probabilistic conditional independencies present in the database. The most famous algorithm of this type is the PC algorithm [CITATION PSp91 \l 1040][CITATION PSp00 \l 1040].

The second method, called search and score, consists of performing a heuristic search through the space of possible structures, using a metric that measures how well each structure can represent the probability distribution of the variables in the database. Several metrics have been proposed in the literature: Bayesian (which include K2 and BDe as particular cases), cross-entropy, AIC, and MDL. K2, the first algorithm of this type, performed a search by departing from a network without links and adding at each step the link leading to the highest score, provided that the score was positive [CITATION GCo91 \l 1040]. The method that proceeds by examining one operation at each step (adding, removing, or inverting a link) is called hill climbing.

6.3 OpenMarkov

The project started in 2002 at the Department of Artificial Intelligence of the Universidad Nacional de Educacion a Distancia (UNED), in Madrid, Spain. Its original name was Carmen [CITATION MAr08 \l 1040], but in 2010 it was renamed as OpenMarkov. They departed from their experience in the construction of Elvira [CITATION The02 \l 1040], an open-source tool begun in 1997 as a joint project of several Spanish universities, but everything in the new program was redesigned and the code of OpenMarkov was built from scratch. The language chosen to develop OpenMarkov was Java, mainly to make it multi-platform. OpenMarkov is able to represent several types of networks, such as Bayesian networks, Markov networks, influence diagrams, LIMIDs [CITATION MAr11 \l 1040], and decision analysis networks (DANs), as well as several types of temporal models: dynamic Bayesian networks, Markov processes with atemporal decisions (MPADs), MDPs, POMDPs, Dec-POMDPs, and dynamic LIMIDs.

Currently it can only evaluate Bayesian networks, influence diagrams, and MPADs. Each network type is defined by a set of constraints, which leads to the possibility of defining new types of networks easily by combining the existing constraints and, if necessary, by adding new ones. Constraints play also an important role in the learning of BNs, as we will discuss below. There are three types of variables in Open-Markov: *finite-states*, *numerical*, and *discretized*. A discretized variable has a finite set of states, each one having an associated numeric interval. The graphical user interface (GUI) is very similar to those of other software tools for PGMs, especially to that of Elvira. It has two working modes: edition and inference. It has been designed for internationalization; currently messages can be displayed in English and Spanish. For further details, see OpenMarkov's web pages and wiki

6.4 Options for learning BNs in OpenMarkov

In this section we describe the main options that OpenMarkov offers for learning BNs interactively.

6.4.1 Using a Model Network

OpenMarkov gives the user the option to use an existing network as a model for the one that will be learned. There are four options. The first is to use the model only to determine the positions of the nodes. When we learn a network from a database we can place the nodes on the screen by dragging them with the mouse, trying to minimize the number of links crossing one another; but if we learn another network from the same database (for example, using different options for the algorithm), we should drag replace

the nodes again. OpenMarkov facilitates this task by placing the nodes in the same positions as in a network built previously. The other three options are whether the algorithm can add, remove, or invert the links present in the model network. They are useful, for example, when we wish that the algorithm preserves all the links in the model network. The first option (i.e., using the model only to place the nodes) is incompatible with the other three, which are compatible with one another. There are other uses of the model network, that we describe below.

6.4.2 Data Processing

Unfortunately data in the databases is usually not suitable to be directly fed to the learning algorithm and has to be preprocessed. Open- Markov offers the following options.

- **Selection of variables.** Usually raw databases contain information that is irrelevant for the model (e.g. the patient's name). OpenMarkov can learn a network that contains all the variables in the database, but it is also possible to tell it to use only those present in the model network. The third possibility is to select the variables one by one from a list.

- **Discretization of numeric variables.** Currently OpenMarkov can only learn BNs with variables having finite states. Therefore, the numeric variables in the database must be discretized before feeding the data to the learning algorithm. OpenMarkov can discretize a variable in different ways. First, the user can indicate a number of intervals and then indicate whether the intervals must have the same width (considering the maximum and the minimum for that variable in the database) or the same frequency (i.e., the number of database registers for every interval will be the same). Second, if the variable is discretized in the model network, its intervals can be used to assign each number in the database to a state. For example, if a variable has three states, “negative", “null", and “positive", with three associated intervals, $(-\infty; 0)$, $[0; 0]$, and $(0;+\infty)$, respectively, these intervals can be used to discretize the values in the database. This way, creating a model network is a way of specifying how numeric variables should be discretized.

- **Imputation of missing values.** Currently OpenMarkov offers only two ways to fill in the gaps in the database: either to ignore every register that contains at least one missing value, or to write the value “missing" in every empty cell, which is then treated as if it were an ordinary value.

6.4.3 List of suggested edits

In OpenMarkov an edit is an atomic modification of a data structure. There are three edits that an interactive learning algorithm can propose: adding, removing, or inverting a link. The list is composed by sorting the edits according to their scores. The hill climbing algorithm computes the scores using the metric selected by the user. The PC algorithm performs many statistical test in which the null hypothesis is that two variables are not correlated given other variables. Roughly speaking, a high p resulting from the test suggests that two variables are conditionally independent, i.e., that a link can be removed. Therefore, the p value can be used as a score to rank the edits, each edit being the removal of a link. Interactive learning is performed by having two windows: one showing the graph of the network and another one showing the proposed edits. The user can select any edit from the list, not necessarily the one having the highest score, and the change will be immediately displayed on the network window. Alternatively, the user can add or remove any link from the graph. In both cases, the scores will be recalculated and a new list will be proposed. Figure 6-1 (in the next page) shows the lists of suggested edits shown during the interactive learning process.

Additional options. There are additional options to control the flow of the algorithm. One of them is tell OpenMarkov to show only the edits having a positive score. Another option is to show only the edits allowed by the constraints associated to the network. For example, a constraint stemming from the definition of the BN is that the graph cannot contain cycles. A constraint that the user can impose is that a node cannot have more than n parents. If the user selects the “Show only allowed edits" option, those incompatible with the constraints will not be shown in the list, even if they have a high score. Finally, the user has the possibility of blocking a certain edit to prevent the system from offering it again and again. Blocked edits can be later unblocked at any moment.

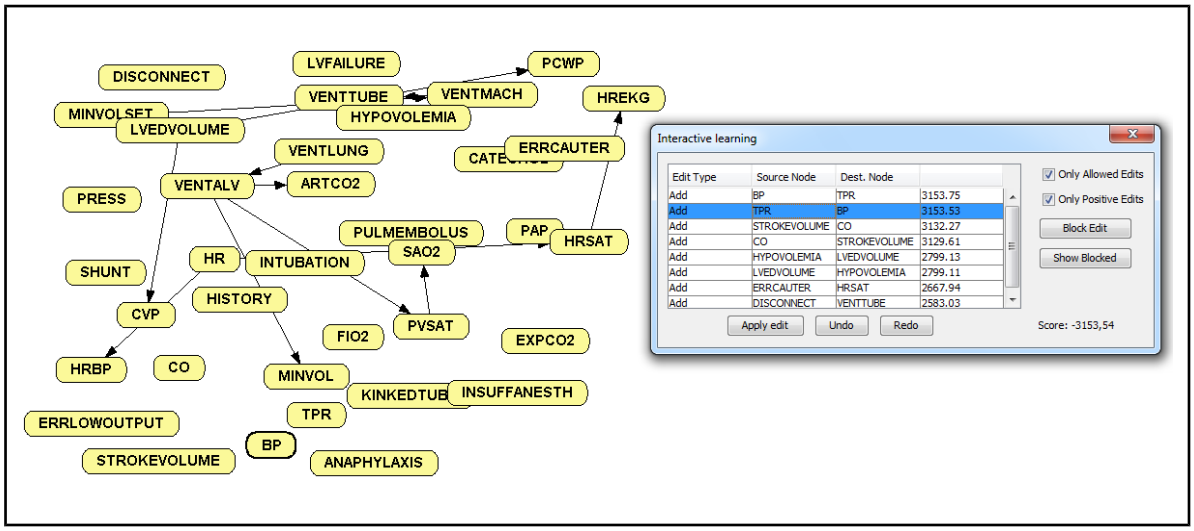


Figure 6-26 A moment of the interactive learning process: list of edits proposed by OpenMarkov and the network being learned.

6.5 Case study

In order to explore the benefits of interactive learning, we study the case of learning the wellknown ALARM network [CITATION IBe89 \l 1040], which has 37 nodes and 46 links. The nodes are classified into three levels. The first level contains diagnostic nodes, which have no predecessors. The second level contains intermediate variables, representing pathophysiological anomalies that cannot be observed directly. The third contains measurement nodes, which represent clinical variables that can be observed or measured, and do not have children. The network contains no link from a lower level to an upper level. Using this network we generated a database containing 10,000 samples and applied the hillclimbing algorithm with the K2 metric. We applied the learning algorithm automatically in OpenMarkov, resulting in a model with 50 links, 13 of which were not in the original network, even though 6 of them were inverted links of the original network.

On the other hand, 9 of the original links were missing in the network learned. Then we learned the network interactively using elementary causal knowledge, according to which we did not accept the addition of any link from a measurement node to an intermediate or a diagnostic node, nor from an intermediate node to a diagnostic node. Figure 6-2 shows an example of a moment in the interactive learning process where the edit with the highest score contravenes the causal knowledge. In this case, we chose to apply the second edit, which produces the same link but in the opposite direction. The resulting net contained 47 links: only 2 of them were not in the original network and only one of the original links was missing. The two links “invented" by the learning algorithm were the last to be added and had such a low score that they might have been detected as spurious by the expert. We also observed that the missing link, from INSUFFANESTH to CATECHOL, has a very weak influence in the original network. This experiment shows that even a portion of very rudimentary knowledge about the domain may lead to a significant improvement in the network built by our interactive learning algorithm.

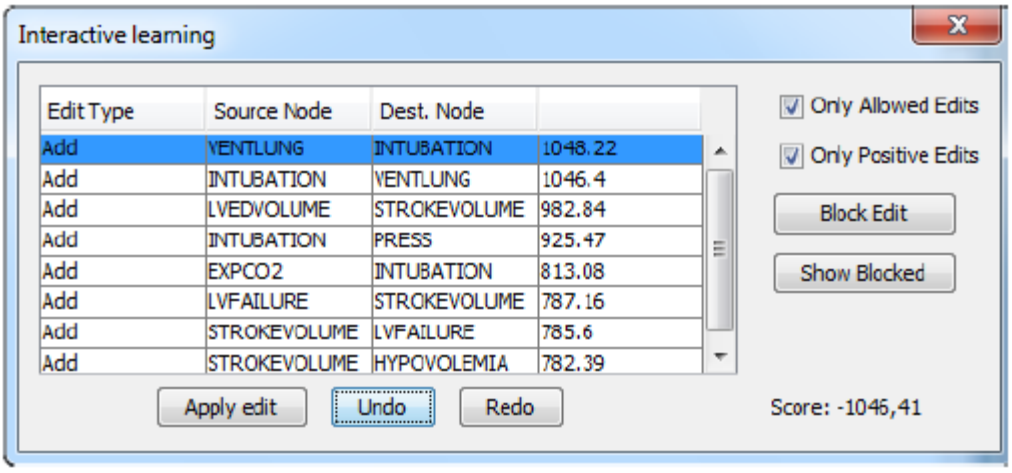


Figure 6-27 The edit suggested at the top of the list contravenes our causal knowledge because VENTLUNG is an intermediate variable and INTUBATION is a diagnostic variable.

6.6 Advantages of this approach

As mentioned in the introduction, a problem of learning algorithms is that they often create spurious links due to small correlations existing in the database. Another problem is that in general the models obtained are not causal, not only because of the inherent limitation of most algorithms, but mainly because the information contained in the database does not permit to distinguish whether X is a direct cause of Y , or X is a cause of Y , or if there is a directed causal path between them involving other variables, or they have a common cause, or there is a selection bias in the database [CITATION CGl99 \l 1040][CITATION MDr03 \l 1040]. OpenMarkov allows human users, who may be experts in their respective areas but novices in the field of probabilistic modelling, to supervise the execution of learning algorithms.

The algorithm proposes some incremental modifications of the network, based on the information contained in the database, and the user has the opportunity to apply some of the changes proposed by the tool or impose others at any moment of the learning process, based on their expertise. Even if this might lead to a lower quality of the network according to the metric, the result might be better from the point of view of users' acceptability, because human experts are reluctant to accept the advice of a machine if they cannot follow its reasoning [CITATION RTe84 \l 1040]. An interactive learning tool might as well be useful for researchers that have developed a new algorithm and wish to trace its execution in order to debug or fine-tune it. This process can be done by inserting in the algorithm a few lines of code that print a trace on the standard output or in a file, but it is much nicer to observe graphically the operations performed by the algorithm, step by step, together with the qualitative information associated with the next modifications that the algorithm has evaluated.

Obviously, this requires that the researchers implement the new functionality (such as a new metric, a new search technique, or a completely novel learning method). OpenMarkov's architecture has been carefully to permit these extensions: each new method can be implemented as a Maven subproject, that OpenMarkov will detect at run time as a plug-in. This way, researchers can extend OpenMarkov dynamically without modifying the official source code. Finally, an interactive learning program may be useful as a pedagogical tool to explain the performance of different algorithms: rather than observing the input and the output, students may follow the progression of the algorithm step by step, understanding why each change was selected, seeing the effects of taking different actions to those proposed by the system and comparing different algorithms.

6.7 Conclusion

In this chapter we have described an interactive learning approach for learning Bayesian networks from databases, which may be very useful for the experts in different application domains, as well as for researchers and students in the field of PGMs. We have shown with a case study that even very rudimentary causal knowledge about the domain may lead to a significant improvement of the network build interactively with a learning algorithm. The main lines for future development in this field would be to represent graphically the strength of the correlation between variables and having richer types of constraints. It would be also useful to show an absolute quality measure of the net rather than the incremental one we currently have, given by the

complexity of the network and the distance between the probability distribution of the network and that of the data. This quality measure could be used to compare the resulting nets of the interactive and non-interactive learning processes. Another research line would be to adapt our approach to learning Bayesian classifiers, a somewhat different problem, as the objective is not to build the network that better represents the probability distribution of the data, but the network that better classifies new cases.

7 CASE STUDY: USER EXPERIENCE OPTIMIZATION IN THE FACETED BROWSING

In this chapter we will analyze in detail our case study. The faceted browsing is a data visualization technique oriented to the management of large amounts of data, commonly used by large enterprises and eCommerce such as Amazon, Netflix and Macy's, who spend most of their IT investments to optimize their solutions in this field. We will see how an innovative approach based on Bayesian networks could lead to an effective optimization of the user experience both in terms of the number and type of facets to display – as the user experience must be as simple and efficient as possible, displaying too many facets of selection would lead to confusion in the choice of the search query – and of inserting a recommendation system that interactively guides the user and allows him to

understand what is going to be displayed from the query he is going to select. In the first part of this chapter we will give a quick overview on the faceted browsing and the tool that we used to handle the visualization of Big Data, later we will analyze in detail the actual optimizations that have been made.

7.1 Introduction: Overview on Faceted Browsing

Figure 7-28 Faceted Search.

Faceted search[CITATION DTu09 \l 1040], also called faceted navigation or faceted browsing, (you can see an example in Figure 7-1) is a technique for accessing information organized according to a faceted classification system, allowing users to explore a collection of information by applying multiple filters. A faceted classification system classifies each information element along multiple explicit dimensions, called facets, enabling the classifications to be accessed and ordered in multiple ways rather than in a single, pre-determined, taxonomic order. Facets correspond to properties of the information elements. They are often derived by analysis of the text of an item using entity extraction techniques or from pre-existing fields in a database such as author, descriptor, language, and format. Thus, existing web-pages, product descriptions or online collections of articles can be augmented with navigational facets.

7.1.1 Where and when to present facets?

A faceted search system responds to a query by returning a set of documents intended to match the search terms, and also provides a set of facets that offer the user directions for query refinement. This faceted search approach, however, does not specify how those two sets (the results and the facets) should be presented to the

user[CITATION DTu09 \l 1040]. An application may choose to present both sets in the same view or may initially present only one set, that is, the results or the facets. Is there an optimal way to make these choices? The short answer is no: different applications and user needs motivate different design choices. What we can do, however, is enumerate a few options and discuss their relative merits. Let us consider interfaces that present both the matching documents and the faceted refinements in the same view.

There are two conventional layouts: placing the facets in a panel to the left of the results and placing the facets directly above the results. An alternative approach is to place the facets directly below the results, but this approach has the drawback that users may not even be aware of the facets unless they scroll to the bottom of the page if the list of results pushes the faceted section below the fold. Placing the results in the center and the facets on the left-hand side makes it more likely that users will see search results and thus focus on them. Users may prefer a view that brings them to the documents as quickly as possible, particularly if they only perform faceted refinement infrequently. For sophisticated users, this vertical layout makes both the results and facets immediately visible. However, this layout may be too subtle for less sophisticated users to notice the availability of the faceted refinement options. Also, such users may be unfamiliar with faceted refinement and may confuse the faceted refinement links with static site navigation links that lose their current query context. Placing the facets above the results makes it easy for users to notice them: they stand between the search box and the results, and they are above the fold. The downside is that placing the facets in such prime real estate means that users may need to make more effort to see the results. However, this effect may have a useful consequence: users may use a faceted search system more productively if they consider ways to elaborate their queries before rushing to the result list for their initial queries – and this strategy promotes the importance of the facets. Another, less common, option is to present results and facets in separate views, for example, each in its own tab.

The main choice in implementing such a design is to force which particular view the user sees first—similar to determining the layout for a view with results and facets presented on the same page. When users cannot see the results from their search, they are likely to make an effort to find them – unless they are so annoyed by the extra work they have to do that they abandon the site. Conversely, they are less likely to make such an effort to find facets that they may not even know exist. Yet another possibility is to make the presentation of results facets a function of the query itself—that is, change the presentation based on the properties of the search results. Faceted refinement can accomplish at least one of two goals: clarification of ambiguous queries and refinement for general ones. For ambiguous queries, it may be appropriate to present the user with a clarification dialog as soon as possible. For example, the search query “intelligence” on a library site might retrieve items about mental ability or information gathering; until the system has established the user’s intended meaning, presenting results is a guessing game. For unambiguous queries, there is less urgency around offering a refinement dialog because the results will at least make sense to the user, even if they are not optimally relevant.

7.1.2 Organizing facets and facet values

In the previous section, we considered the problem of information overload from a back-end perspective: how do we prune the number of facets or facet values that we present to users? In this section, we consider the corresponding front-end problem: how do we organize the information that we do present? We approach the challenge of information overload in two steps: reducing the number of facets and values presented and then

organizing them as effectively as possible. There are three general strategies for organizing facets, and these are similar to the strategies discussed in the previous chapter:

- Use a static order that does not change as the user navigates.
- Dynamically rank the order of presentation of facets based on their estimated utility to the user.
- Organize similar or related facets into groups.

The choice of using a static facet order vs. ranking facets is an interesting trade-off. On one hand, a static order has the advantage or reinforcing the user's mental model—because the user will always see the same facets in the same order. This strategy works best when the number of facets is small in that all facets are visible at all times. On the other hand, a static ordering is less effective when the number of facets is too large to be displayed at once or when some facets only apply to particular query contexts. For example, on an e-commerce site that sells consumer electronics, some facets only apply to small subsets of the product catalog, such as wattage for audio speakers or megapixels for digital cameras. Those facets should only be displayed when the user is looking at narrow result sets. In general, the same kinds of utility measures used for filtering can also be used for ranking. Grouping related facets, as in ACM Digital Library's grouping related people, publications, and conferences (Figure 7.3), makes it possible to include more facets in a single display using less space on the page because the groups can be expanded and collapsed as desired by the user. Rich internet applications, designed using such technologies as AJAX (Asynchronous JavaScript and XML) make it easier to implement such an approach that still offers users a highly responsive interface. Similar strategies apply for presenting the facet values:

- Use a static order that is independent of the query context.
- Rank facets based on a utility measure.
- Present hierarchical facet values progressively, that is, one level of the hierarchy at a time.

This last strategy is an option even when the facet is not hierarchical. We can create an artificial hierarchy in which no node has more than a tractable number of children, for example, 5 or 10. For example, consider a facet with a large set of strings as values, such as book authors. We can divide up them up by last name, splitting them at the top level as A–E, F–K, L–P, Q–U, and V–Z. The A–E node can be split into A, B, C, D, and E; A split into Aa–Ae, Af–Ak, etc.; and so forth until no node has more than five authors as (leaf) children. The design of such an artificial hierarchy requires some care to mitigate the trade-off between excessive depth and excessive fan-out at each node. We can take a similar approach to create a hierarchy of ranges for numerical values or for facets whose values can be clustered based on a similarity measure.

7.1.3 The search box

Although our discussion of faceted search has focused on the use of facets for refinements, it is important to remember that faceted search is still a type of search – and that often the entry point into a faceted search system is the search box. Without the search box, we would only have a faceted navigation system—a useful interface for many applications but too limited to enjoy the broad success of faceted search. Combining free-text search

and faceted refinement is powerful: it allows users to create semistructured queries and thus access structured and unstructured content. However, the search box also raises significant design challenges for application developers. The designer of a faceted search system must make a number of choices about how the search box behaves:

- Should a search query adhere to the current query filters?
- Should search look at all of the text in each document, or should search be restricted to specific fields?
- How should the search handle multiword queries by default? Should the words be combined as an OR (i.e., match any word), as an AND (i.e., match all words), or as a phrase (i.e., the words must all occur in a document and in that exact sequence), and should the user be given a choice in this matter?
- Should search queries be subject to query expansion, such as matching words that are variants of query terms?
- Should systems present multiple search boxes, a parameterizable search box, or an advanced search interface?

These are open-ended questions and only represent a subset of the questions about search behavior that face designers of faceted search applications. We will try to supply some answers—or, at least, guidance.

First, let us consider the question of whether the search query should respect the current query filters. In the most common use case for faceted search, a user initially enters a free-text search query and then follows it up by one or more refinements using the facets. For example, a user types in “digital cameras” and refines on a specific megapixel range. But how do we handle deviations from this common case, for example, when the user first narrows the document collection by selecting a facet value and then performs a free-text search? Does the text search adhere to the faceted refinement or start a new query from scratch? The conventional and probably safest approach is for free-text search to default to clearing all other filters or to offer users the options to search within the current results, for example, by clicking on a check box indicating that the user is explicitly choosing to search only within the set of results that is currently being viewed.

Now let us consider the question of whether we should match a search query against the full document text or only a restricted set of text fields. A common approach for search engines is to perform search against full text and to rely on relevance ranking to push more relevant results to the top of the results. Although this approach may work well in a conventional search engine, it can undermine the effectiveness of faceted search. Faceted search builds on a set retrieval model: the faceted refinements reflect all of the results not just the results that the system judges to be most relevant. If most of the results are not relevant, then the faceted refinements may not be especially useful, especially if they are presented with counts indicating their distribution over the result set. An alternative approach is for the default behavior to err on the side of precision, for example, searching only against the title field unless the user explicitly asks to include other fields. The key design question is whether the benefit of increased precision cost usually outweighs the cost of decreased recall. It is also possible to hedge, to obtain search results from a broader search query that favors recall, and to derive the faceted refinements from a narrower search query that favors precision.

The search box can also serve as a way for the user to search the set of facets, not just the documents themselves. An advantage of this approach is that the set of documents assigned a particular facet value is often a more accurate result set than the set of documents containing those words in their text. It is even possible to search against the set of combinations of facet values. Multiword search queries and query expansion also raise issues of precision and recall. However, because of their familiarity with web search engines, most users have become accustomed to search engines that interpret multiword search queries as an unordered conjunction (an AND, not an OR), for example, a search for faceted navigation returns documents containing both words but not necessarily in that order or even next to each other. While we should not be fatalistic about conventions, we must recognize that flouting them will incur some amount of user confusion. The conventions for query expansion are less established, but most users expect, at a minimum, that they do not need to worry whether they use the singular or plural form of a noun (i.e., that both will return the same results). More aggressive query expansion (e.g., employing a thesaurus to obtain additional matches for words related to the query terms) is again a precision/recall trade-off.

More importantly, it calls for transparency to avoid confusing users with unexpected and unexplained results. Any expansion that is unintuitive to a user is not worth the risk of confusing users and thus undermining their faith in the system. Finally, there is the question of whether to offer users multiple search boxes, a parameterizable search box, or an advanced search interface. There are no hard and fast rules here, but multiple search boxes with different behaviors have the potential to confuse users. A few users will configure a parameterizable search box, but most will never change the default search behavior. Hence, it is critical that the default search behavior be reasonable. Similarly, whereas a minority of users will appreciate the opportunity to use an advanced (typically parametric) search interface, many will never even discover it exists. To avoid confusing the majority of users, many successful retrieval applications place their advanced search interface on a separate page.

7.1.4 Multiple selection from a facet

The most common use case for faceted search or navigation is to select at most one value per facet, but there are at least two ways from which a user might select multiple values from the same facet:

- Disjunctive (OR) selection. Selecting a range (e.g., a price or date range) may be a kind of disjunctive selection, depending how the values are represented.
- Conjunctive (AND) selection.

The design challenge is to communicate to users whether selecting multiple values from a particular facet is disjunctive or conjunctive – particularly if the site offers both behaviors. Users are notoriously bad at inferring Boolean logic from subtle cues. It is important to use an interface that not only is self-consistent but also adheres to familiar conventions.

There are fewer interfaces that allow conjunctive selection from the same facet, but a convention for those that do is to present the selections as ordinary links. The approach may make the user think that he or she is drilling

down a hierarchy, but fortunately that misinterpretation is consistent with the narrowing effect of conjunctive selection. Perhaps most importantly, we urge caution in combing disjunctive and conjunctive selection in the same interface. Users who can understand such a complex process will be better served by the ability to construct Boolean queries at a command line. A rule of thumb is that facets that are typically singly assigned to documents (e.g., brand, document type) work well with disjunctive selection, whereas facets that are often multiply assigned to documents (e.g., consumer electronics features, topic) work well with conjunctive selection.

7.1.5 Challenges ahead: Optimization of the User Experience

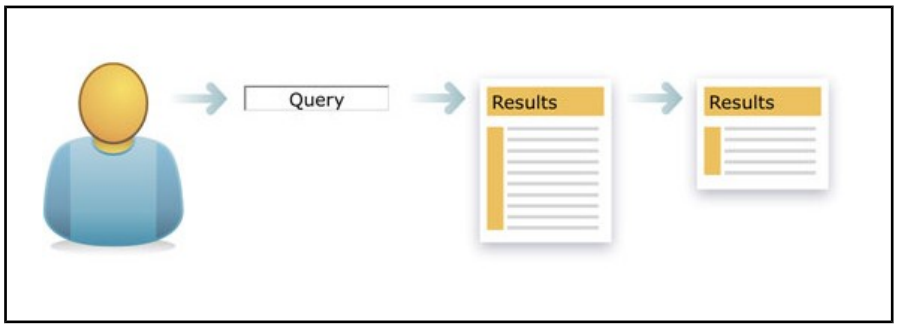


Figure 7-29 Faceted Browsing Design Pattern.

The power of faceted search can overwhelm and confuse users if it is implemented with a poor design. Choosing the correct layout of facets and results in an interface can be a trade-off between the work required for users to see results and the likelihood they notice the facets. For ambiguous queries, users may benefit from a facet-driven clarification dialog. Strategies to avoid information overload by filtering facets and facet values also offer ways to rank and organize them.

Users generally expect that initiating a new free-text search will clear current query filters, but some applications provide the option to search within current results. A number of decisions about search behavior involve a precision/recall trade-off: consider computing results to favor recall but computing the utility of facets and facet values based on a narrower search query that favors precision. An interface allowing multiple selections within a single facet should not only be selfconsistent but adhere to familiar conventions. Consider the use of design patterns to take a holistic approach and learn from the collective wisdom of practitioners.

7.2 Implementing Faceted Browsing with Apache SOLR

Solr is an open source enterprise search platform from the Apache Lucene project. Its major features include powerful full-text search, hit highlighting, faceted search, near real-time indexing, dynamic clustering, database

integration, rich document (e.g., Word, PDF) handling, and geospatial search. Solr is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized conFiguretion and more. Solr powers the search and navigation features of many of the world's largest internet sites like Macy's and Netflix.

Solr is written in Java and runs as a standalone full-text search server within a servlet container such as Jetty. Solr uses the Lucene Java search library at its core for full-text indexing and search, and has REST-like HTTP/XML and JSON APIs that make it easy to use from virtually any programming language. Solr's powerful external conFiguretion allows it to be tailored to almost any type of application without Java coding, and it has an extensive plugin architecture when more advanced customization is required.

Solr runs by default on a light servlet container like Jetty but, in order to enable a solution that can handle the visualization of Big Data, we need to change it with a more powerful servlet container. So we need to integrate those applications: Solr, Tomcat, Apache Zookeeper, Hadoop File System (HDFS).

7.2.1 Overview on Apache Solr: Sharding and Replication

The best way to manage Big Data with Solr is troughed sharding and replication. In detail, sharding is a type of data partitioning that separates very large data into smaller, faster, more easily managed parts called *data shards*. Technically, sharding is a synonym for horizontal partitioning. In practice, the term is often used to refer to any database partitioning that is meant to make a very large database more manageable. The governing concept behind sharding is based on the idea that as the size of a database and the number of transactions per unit of time made on the database increase linearly, the response time for querying the database increases exponentially.

Additionally, the costs of creating and maintaining a very large database in one place can increase exponentially because the database will require high-end computers. In contrast, data shards can be distributed across a number of much less expensive commodity servers. On the other side, replication is the process of creating and managing duplicate versions of a database. Replication not only copies a database but also synchronizes a set of replicas so that changes made to one replica are reflected in all the others. In order to provide high availability, I can create replicas, or copies of each shard that run in parallel with the main core for that shard. The architecture consists of the original shards, which are called the leaders, and their replicas, which contain the same data but let the leader handle all of the administrative tasks such as making sure data goes to all of the places it should go. This way, if one copy of the shard goes down, the data is still available and the cluster can continue to function.

In this chapter we will omit the design phase of installation, conFiguretion and analysis of the Solr's implementation details as they are not of particular interest in our case study. In the next sections we explain instead how we have optimized the faceted browsing both theoretically and describing some implementation details of our open source solution.

7.3 Selecting Top-K Facets for High-Dimensional Structured Data

Searching and browsing are the two typical ways by which users locate items of interest. At present, faceted interface has become one of the most widely used interface because it supports integrated, seamless searching and browsing, with very high degree of usability. The usability of a faceted interface depends quite a lot on what facets and facet-values are shown and how they are organized. Thus far, faceted interface are mostly created manually or semi-automatically. If a dataset has few facets and each facet has few facet-values, then designing a faceted interface is quite easy because it has very small design choices. However, if there are many facets or facets have many facet-values, then the faceted interface designer has to manually select the appropriate subset of facets that should be shown and come up with a suitable ranking function by which the top-k facet values should be selected from facets that have many facet-values. In this section, we show the difficulties that users face while designing a faceted interface for high-dimensional, structured data, especially related to identifying the top-k facets. Generally, designer of a search interface is not the same person who has generated the data or has good domain knowledge, and thus it requires lot of feedback between the designer and the domain expert on what to and how to present the data to the world.

We propose a system that will make the process of designing a faceted interface more automated, and thus reduce the amount of feedback that is otherwise required from the domain expert. In this paper, we propose three types of facet selection algorithms, namely *no-feedback*, *one-time feedback* and *iterative feedback*, to select the top-k facets. Depending on how precisely the designer knows what information they want to share, they would find top-k facets recommended by one of these three facet selection algorithm more suitable.

7.3.1 Introduction

Faceted navigation is one of the most important breakthroughs in modern website design. Almost all e-commerce sites are publishing their huge databases through faceted search interface. However, since faceted interfaces are relatively new, they have various pitfalls and are currently being addressed by many researchers in both IR [CITATION MHe06 \l 1040] and Database[CITATION AKa10 \l 1040][CITATION DDa08 \l 1040] communities. In "*Designing search: UX strategies for ecommerce success"*, [CITATION GNu11 \l 1040] Nudelman gives a case study of how Office Depot had redesigned their search interface to support faceted search, but due to wrong design choices, Office Depot's site had temporarily become less usable.

Designing a faceted interface requires lot of manual decisions, and each of these decisions can greatly influence their effectiveness. Faceted navigation is a type of categorization technique. Prior to faceted navigation, people used to design single-concept hierarchies and categorize the results according to those hierarchies. Clearly, its a great pain to manually de- sign such hierarchies for huge amount of data, especially when the data has many dimensions. Moreover, single-concept hierarchies are not so usable because the groupings are quite subjective, based on the hierarchy designer's choices, and may not be aligned with user's preferences. On the other hand, faceted navigation is so successful because it allows users to place selection conditions independently from any of the available facets. The designer has significantly less amount of burden to figure out the right hierarchical

sequence to organize their data. In "*Faceted metadata for image search and browsing*", [CITATION MAH06 \l 1040] it has been shown through user studies that faceted navigation is more usable as compared to clustering or single-concept categorization. Searching and browsing systems vary in the amount of manual effort that is required to design those systems. For example, the process of designing a keyword search interface is quite automated because the designer just needs to feed in all the available data and just select the appropriate ranking function. Similarly, if the data is low-dimensional and has few values in each dimension, then the process of creating a faceted interface is quite automated because the designer has to make few design choices.

However, if there are many facets or facets have many facet-values, then the effectiveness of a particular faceted interface is quite dependent on designer's choices. Since the interface designers are often not the domain experts, it is very challenging for them to manually figure out the best design choices and thus it requires too many iterative feedback loops between the domain expert and the interface designer. For example, a Bioinformatician may hire an interface designer to publish her research findings, and the designer might have very limited knowledge of Bioinformatics. Since an interface has long-term impac tand is based on subjective notion of importance, it is ideal to make the design choices by using combined effort of computers and domain-experts, where the system helps the domain expert by suggesting good set of recommendations. To create faceted interface for structured databases, one can consider the attributes as facets and the attribute values (discretized for numerical attributes) as facet-values. Although, there are many concerns in designing a faceted interface, in this paper, we focus on the following two concerns:

- **Top-k facets in high-dimensional dataset**: Many applications, such as scientific databases, e-commerce etc., have very high-dimensional data. Although more dimensions are desirable because they give deeper insight of data, but they also leads to information overload. If all the dimensions as are shown as facets, then it will make the faceted interface cluttered. For any dataset, each dimension explains certain aspect of the data. Based on what information a user wants to share, she can chose the appropriate subset of dimensions. However, it is very hard for a user to manually identify the subset of relevant dimensions because of lack of domain knowledge and also lack of knowledge of how different dimensions interact with each other. For example, if we see the keyword search technique that is commonly used in IR or databases, then in such search tools the user specifies a few search keywords, and then the system identifies all the documents or tuples that might be of users interest. Similarly, in designing a faceted interface, the designer can provide some amount of information on what kind of information they want to share, and then the system should help the user search the top-k facets that would be most relevant.

- **Organizing facets for better visual effect**: Grouping related information is often useful because it reduces the amount of back-and-forth browsing that is required by the user. For example, in the problem of Market Basket analysis in data mining, the goal is to co-locate related items, so that users can easily look and compare multiple related items. Similarly, grouping related facets can reduce the amount of back-and-forth browsing required by the users. If related facets are placed adjacently, then the user can easily see the effect of selecting the values on one facet on the related facets.

7.3.2 Faceted Interface, User Interface

Problem Statement:

Given a structured dataset D with n features $F = \{f_1, f_2, \dots, f_n\}$. Our goal is to design a system that would assist users to select a subset S of k features from F, such that S is the set of top-K most informative facets for D.

Although in designing a faceted interface it is often considered desirable to have a set of facets which are independent, but independence alone is not a sufficient metric. Importance is often determined by what is considered important by many users. For example, in a used-car dataset, although features such as `Year` and `Mileage` are quite correlated, but both of them are considered to be an important feature because they are very crucial in determining the feature `Price`, which is one of the most important feature for all users. In this section, we present three classes of facet selection algorithms that can be used to select the top-k facets. These algorithms differ in terms of the input feedback received from the domain expert, and we call these three classes as: **(a) No-feedback**, **(b) One-time feedback**, and **(c) Iterative feedback**. In no-feedback type of facet selection algorithms, the top-k facets are selected without any feedback from the user. These algorithms are given as input the set of all n features F, and they return the top-k facet set S without any user input.

- In one-time feedback type of facet selection algorithms, the input is the set of all n features F and a subset of preferred features P, and the output is a set of k features that are most informative given P. These algorithms are very similar to keyword search algorithms, where users specify a small number of keywords and the algorithms return a ranked list of documents or tuples that are most relevant for the given search keywords.

- In one-time feedback type of algorithm, our assumption is that although the domain cannot identify the exhaustive set of facets that are important, but can provide a small subset of facet that are relevant for the type information that the user wants to share, and then we automatically identify other facets that might also be relevant for the user.

- In iterative feedback type of facet selection algorithm, the user is does not give any subset of preferred features, but the user is willing to go through the process of identifying the top-k facets along with the system.

In these methods, the system recommends the facets in a sequence, and the user can select whether they like certain facet or not. In this section, instead of finding the top-K subset by just using some unsupervised feature selection algorithm, our goal is to take the domain expert in the loop to determine

the top-K facets. Different facets represent different aspects of a data and all the diverse aspects may not be equally important to be shown as possible facets. We design an iterative system, where the system tries to identify the different groups of facets that describe specific aspects of the data and we show this information to the domain expert. Based on the feedback of the domain expert, we try to again generate the facet groupings so that it more closely matches the need of the domain expert. Our goal is to present the facets in such a manner that it would enable the system designer to easily make decisions on how to organize the facets. Since the display-space is limited, if there are facets that have many facet-values, we need ways to rank the facet-values so that we can show the top-M most relevant facet-values.

Often the technique that is used for ranking is frequency count. Although frequency count is an important strategy because it gives the most promising values based on user's current choices. But it misses the important strategy that has been used by all search engines for ranking: top-M most similar items and top-M most diverse items. When the users are not specific about their goal, then diversity is preferred because it satisfies the wide search range. On the other hand, when the users have more precise search goal, then similarity is preferred because user is interested to see the top-M most similar items to what the user has already provided. Doing simple frequency based ranking is not going to give either of these rankings. In this section, our goal is to design a system that automatically infers how precise the user's current search is for each facet and then accordingly varies the amount of diversity or similarity in the top-M values. In order to determine, the amount of diversity or similarity that is suitable for a particular facet based on user's current selections, we use the following expression:

$$FacetPreference(f) = \begin{cases} Diversity, & \text{if } PrScore \leq \delta \\ Similarity, & \text{otherwise} \end{cases}$$

Given a facet f, we infer the type of facet-values that we should select from that facet using the above equation. We look at the values that are selected from f or facets that are closely related f, and then compute a score called PrScore (precision score) that measures how precisely the user has selected values from f. If PrScore is less than a given thresh-old parameter δ, then we compute the Top-M diverse values, or else we select the Top-M similar values.

User Interface. Since faceted interface is quite new, extensive usability studies have not yet been for all design aspects. At present, most faceted interfaces display only few facets, and thus the ordering of facets does not make a crucial difference. However, if there are many facets, then showing the related facets adjacently would be quite useful because when a user selects certain values from a facet, the facets on which their would be more effect of the selection would be adjacent to the selected facet. By placing related facets we not only reduce the panning and scrolling burden of the user, but we can also give an insight to the user of how the facets are related with each other. For example, we can keep the facets `Body Type` and `#Doors` adjacent to each other. In this

section, we do not go into the detail of how to organize the facets and evaluate their effects, but we show how to compute the related facets.

Facet values can be selected in two ways: single and multiple selection. Both single and multiple selection have their respective pros-and-cons. Most faceted interfaces support only one selection per facet dimension. For example, for the facet `Color`, a user can select only one value such as `Color: Red`. Although single selection is perfectly adequate for many tasks, many sites also support multiple selection where users can select multiple values from the same facet. For example, `Color:Red` OR *Color*:Blue. Multiple selection is useful because users can compensate for the lack of retailer's perfectly normalized data simply by selecting multiple values. For single-selection facets the values are typically ordered by count, but in multiple-selection facets they are often ordered alphabetically. Sorting by count is useful because to some extent it gives the most important values at the top of ordered list. But if we want to allow multiple-selection, then the facet-values are showed in static manner and their relative position cannot be changed. By not reordering the facet-values based on their result count, we reduce the usability of the interface because the user has to now manually scan through long lists to see which values have high-frequency count. In this paper, we suggest a modified version of multiple-selection that has useful ranking function like single value selection and also allows selecting multiple facet-values. Let's assume that we can show at most L facet-values from each facet. We divide the section where facet-values are shown for each facet in two parts: selected and recommended facet-values. If the user has selected M values from a facet, then we show those M values in the selected subsection and the remaining (L − M) values in the recommended subsection. The facet-values that we show in the recommended subsection is based on the values that the user has selected in the facet or the facets that are closely related with the given facet. We diverse or similar facet-values based on the automatically derived FacetPreference for that facet.

7.3.3 Computing Top-K Facets

Naïve Solution. Ideally, in a faceted interface the goal is to select facets that are independent of each other. We can use an unsupervised feature selection algorithm to find the K most independent subset features. Clearly, this is not a very desirable solution. Because this like a programming IDE which fills in the code without the option for the user to select the right option. Although suggestions from a system is desirable, but it should be based on what the user wants. The user has partial understanding of what is their goal and the system has information of what is good for all possible users. Can we design the system so that the system takes small amount of input from the user and gives appropriate suggestions that are more customized for the user. To do this we next give various iterative solutions to the problem.

Seeded-Feature Selection. In this feature selection, we assume that the user has provided some S facets as important, where S << K, and our goal is to find the (K − L) other features that are most useful given the S features from the user. We need to modify the existing feature selection algorithms so that they select the Top-K features using these already given set of fixed features. I don't think there is an existing algorithm to do this, so may have to suggestion new feature selection algorithm to do this. In this we are assuming that although the user can partially say what is important for the user, but user would not be able to explicitly list down all the facets that would be relevant. Our algorithm would help the user to avoid missing important facets. For example, in the

used car dataset the user may say that `Price`, `Make` are important facets. We can then ask the system to generate two more facets that the system feels is more important based on these choices by the user. We can give different seeds and see what are the suggested features by the user. In this we are assuming that the user would be able to give in the seed form all the information that the user feels are important, and our goal is to just figure other information that embellishes the seed like information that is given by the user.

Iterative-Feature Selection. In the seeded-feature selection algorithm we assume that the user is able to give us a summary set of facets that the user typically considers to be important. We select other facets based on whatever information the user has provided to us in the beginning. Clearly, this is also not a very desirable solution because it is very unlikely that users would have exhaustive understanding of their own data beginning. Rather as they look at the data more closely through numerous data analysis tools, they can get more insights and then accordingly make decisions. In the iterative-feature selection algorithms, we show to the user how the features are dependent on each other in the form of graphs. User can give suggestions on the facets that they consider important on the graph. Our system would take the user's suggestion and try to find out other relevant features. We would also show to the users in the condensed graph view the aspects of data that the user has not selected from the graph in a concise manner. If the user feels that certain set of features has been missed, then they can be again selected at this point. Given a dataset with many features, it is almost impossible for anyone, including the domain experts, to list down the dependency between various features.

For example, con- sider a simple dataset of used-cars that has features, such as, `Make`, `Model`, `Year`, `Price`, `Mileage`, ... Although many people have quite a good background knowledge of this data, but still they would find it hard to explicitly list down the dependencies between these features, as shown in Figure 7-3. In the directed graph shown in Figure 7-3, there is an edge between two nodes if we can infer the value of one node with high probability, if we know the value of the other node. For example, if we know the `#Doors` in a car, then we can easily infer what is its Body Type. Similarly, we can verify other intuitive dependencies, such as feature of *Price* is primarily dependent on `Year`, `#Engine`, and feature `Model` is primarily dependent on `Make`, `Body Type`. We can also see that feature such as `Color` is not related with any other feature.

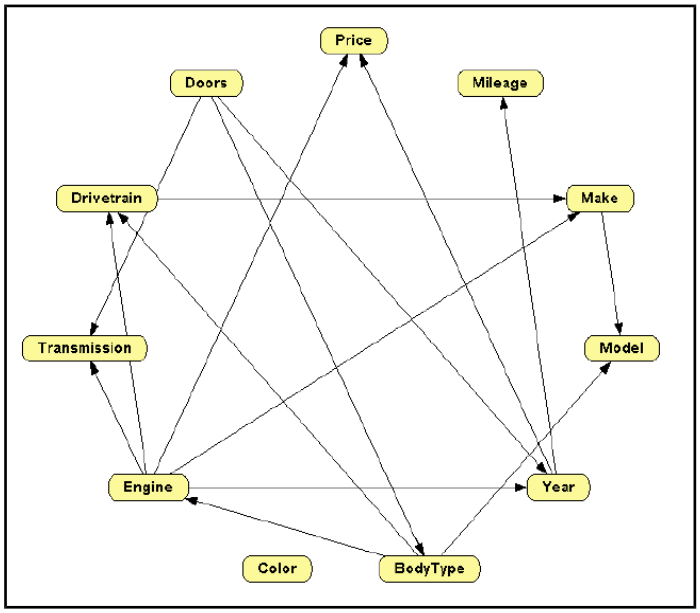


Figure 7-30 Dependency between car features shown using OpenMarkov GUI.

However, we can see that in most applications, just independence is not the typical criteria by which people typically select facets. For example, The features that are shown in Figure 7-3 are the common features that are shown as facets by most car related e-commerce sites. Although a car has more than 100 features, yet all sites show these facets because these are the once that are most crucial for the users. If we consider all the features that are their in the cars, then we can surely find many subset of features that are quite independent of each other and can be easily obtained using existing feature selection algorithms, but none of the e-commerce sites list those features as possible facets. Thus, although it is desirable that facets should explain independent aspects of data, but it is not the only concern. Finding top-K facets is just like finding the top-ranked pages on the web. Importance of the web pages is not determined just by seeing their diversity, but also depends on the other web pages with which they are linked. For example, since feature `#Engine Cyl` is quite closely related to feature `Price`, and we all know that *Price* is an important feature. So, we find that all the features that are important in determining price are used as facets, although they may have less independence as compared to some other features in the car, such as `Audio`, `Seats` etc.

7.3.4 Conclusion

Finding dependency between attributes to do tasks such as query optimization, approximate query answering, query profiling etc. This is done by creating histograms, synopses etc. They also create dependency models. We can use these techniques that have been used to analyze the data to automatically infer the right subset of facets [CITATION ADe01 \l 1040][CITATION PBr03 \l 1040]. We will do our experiments on car dataset as the real data. We can use multiple synthetic dataset that can be generated using Bayesian Network datasets.

7.4 Query Recomandation System

In this section we will describe the second user experience optimization implemented during the research project: In the selection panel of Solr we inserted an interactive recommendation system which can guide the user to choose the query that most fits its research requirements. Because of the increasing of the number of products to display, and of the complexity of the characteristics that define the products, the user's search experience can become complicated. If you do not have a complete knowledge of the product you will be looking for, the facets may seem interchangeable for inexperienced users. The user who chooses one facet instead of another will not be able to figure out where this choice will lead him. To solve this problem and to make the process of query selection the most simple and efficient as possible we have developed an interactive system of recommendation.

7.4.1 Dynamic Summary and Query Recomandation System: Analysis

When the user hovers over a facet in the Solr's selection panel a javascript function will communicate to a servlet the current query and the facet that the user intends to select hovering over it, then the servlet calls our implementation of the OpenMarkov's API, that, after reading these information, reads and loads the Bayesian network previously calculated using the OpenMarkov GUI and calculates the global probabilities twice, one for the current query inserting each value in the `PRE` probabilities matrix and and one for the query to which the facet the user was hovering is added as an evidence in the Bayesian network, and the recalculated values will be inserted into the `POST` probabilities matrix.

Then, for each value of probability it will be calculated the standard deviation between the value in the `PRE` matrix and the value in the `POST`. Now we can define if a certain facet can be added into the category `ADDED`, `UNALTERED` or `DELETED`. Obtained the three arrays of `ADDED`, `UNALTERED` and `DELETED`, we order them by the greatest probability to the lowest, and for each array the algorithm selects the top 5 values that will be sent to javascript function through the servlet and will be displayed in a HTML tabular format in a popup, as is shown in Figure 7-4.

In this way the user is facilitated in his process of search because at each step of hovering on each facet he will have real-time knowledge of how the eventual selection will affect the search. Furthermore, it was also introduced an optimization of the resultset, the javascript function will also send the informations about the current query to another servlet, which, using the API of OpenMarkov, calculates the 7 most significant facets for that query. Using this approach also the resultset provided by the user's query will be optimized with a dynamic summary: For each item that has to be displayed the 7 most significant facets for that query and other 3 random facets – in order to add the element of randomness between the various features of a product description – will be shown. In the next section we will see in detail some parts of the implementation of our solution.

Figure 7-31 Query Recommendation System for a mushrooms dataset.

7.4.2 Implementation & Coding

JavaScript Class: MULTIFACET.js

function showPopup(mouseIn)

```
var string_to_servlet = "NUMBER"+choices.length+"PRE"+choices+"POST"+variable_velocity;
pageRequest.send(string_to_servlet);
var pageStringHtml = pageRequest.responseText;
popup = window.open("","","width="+w+",height="+h+",top="+t+",left="+l+",scrollbars=no menubar=no status=no titlebar=no location=no");
popup.document.write(pageStringHtml);
```

Function called from facet_field.vm that sends a String to the servlet and it makes a popup that is computed dynamically with Open Markov. Call java servlet and return a html string to show the summmary.

function dynamicSummary(value1, value2, value3)

```
var pageRequest = new XMLHttpRequest();
pageRequest.open("POST", url_servlet, false);
var string_to_servlet1 = "NUMBER"+"3"+"PRE";
string_to_servlet1 += facet1 + tag2 + facet2 + tag2 + facet3 + tag2;
string_to_servlet1 += "POST" + facet1;
var string_to_servlet2 = string_to_servlet1.replace(/ /g,'');
pageRequest.send(string_to_servlet2);
var summary = pageRequest.responseText;
document.write(summary);
```

Shows a dynamic summary of each item made with Open Markov. The top 7 values show are dynamic and the last 3 are static (random).

Java Servlet: OpenMarkovRecomandationSystem.java

```
private double[][] preProbabilities;
private double[][] postProbabilities;
private double[] preAverage;
```

```
private double[] postAverage;
private double[] preStandardDeviation;
private double[] postStandardDeviation;
private double[] meanStandardDeviation;
private ArrayList<String> unalteredKey;
private ArrayList<Double> unalteredValue;
private ArrayList<String> addedKey;
private ArrayList<Double> addedValue;
private ArrayList<String> deletedKey;
```

Some of the constants used in OpenMarkov API for the pre & post probabilities computation.

```
// Open the file containing the network
InputStream file = getClass().getClassLoader().getResourceAsStream(bayesNetworkName);
// Load the Bayesian network
 PGMXReader pgmxReader = new PGMXReader();
ProbNet probNet = pgmxReader.loadProbNet(file, bayesNetworkName).getProbNet();
// Create an ArrayList of ALL Bayesian network nodes
ArrayList<Variable> bayesianNodes = new ArrayList<Variable>();
bayesianNodes = (ArrayList<Variable>) probNet.getVariables();
```

Initialization of the variables of the OpenMarkov API.

```
if (state.equals("pre"))
{
        preProbabilities = new double[variablesOfInterest.size()][maxNumValues];
        preAverage = new double[variablesOfInterest.size()];
        preStandardDeviation = new double[variablesOfInterest.size()];
        double[] values;
        double preDelta = 0;
        NameField = new Variable[variablesOfInterest.size()];
        NameValue = new String[variablesOfInterest.size()][maxNumValues];
        for (int i = 0; i < variablesOfInterest.size(); i++) {
           for (int j = 0; j < maxNumValues; j++){
                preProbabilities[i][j] = 0;
                NameValue[i][j] = "";
              }
         }
}
```

Construction of a matrix containing all the preProbabilities and of another matrix that contains each name value of each field. The two matrix has the same position so I can know the probability of a field_value just knowing the position of the NameValue matrix.

```
if (state.equals("post")) {
```

```
        postProbabilities = new double[variablesOfInterest.size()][maxNumValues];
        postAverage = new double[variablesOfInterest.size()];
        postStandardDeviation = new double[variablesOfInterest.size()];
        meanStandardDeviation = new double[variablesOfInterest.size()];
        double[] values;
        double postDelta = 0;
        for (int i = 0; i < variablesOfInterest.size(); i++) {
           for (int j = 0; j < maxNumValues; j++) {
              postProbabilities[i][j] = 0;
        }
}
```

Construction of a matrix containing all the postProbabilities and of another matrix that contains each name value of each field. The two matrix has the same position so I can know the probability of a field_value just knowing the position of the NameValue matrix.

```
for (int i = 0; i < variablesOfInterest.size(); i++) {
            Variable variable = variablesOfInterest.get(i);
            TablePotential posteriorProbabilitiesPotential = posteriorProbabilities.get(variable);
            values = posteriorProbabilitiesPotential.getValues();
            for (int j = 0; j < values.length; j++) {
               if (j == 0) {
                  postAverage[i] = 0;
               }
               postProbabilities[i][j] = values[j];
               postAverage[i] += values[j];
            }
            postAverage[i] = postAverage[i] / values.length;
            for (int j = 0; j < values.length; j++) {
               postDelta = values[j] - postAverage[i];
               postStandardDeviation[i] += Math.pow(postDelta, 2);
            }
            postStandardDeviation[i] = Math.sqrt(postStandardDeviation[i] / (values.length - 1));
            /* Compute also the mean from the two standard deviation for each field*/
            meanStandardDeviation[i] = (preStandardDeviation[i] + postStandardDeviation[i]) / 2;
        }
}
```

Set the two previous matrix and calculate the post mean and the post standard deviation for each field.

```
ArrayList<String> unalteredKey = new ArrayList<String>();
ArrayList<Double> unalteredValue = new ArrayList<Double>();
ArrayList<String> addedKey = new ArrayList<String>();
ArrayList<Double> addedValue = new ArrayList<Double>();
ArrayList<String> deletedKey = new ArrayList<String>();
ArrayList<Double> deletedValue = new ArrayList<Double>();
```

Inizialitazion of the Query Recomandation System categories' variables.

```
for (int i = 0; i < postProbabilities.length; i++) {
   for (int j = 0; j < postProbabilities[i].length; j++) {
      if (((postProbabilities[i][j] - preProbabilities[i][j]) < meanStandardDeviation[i])
            && ((preProbabilities[i][j] - postProbabilities[i][j]) < meanStandardDeviation[i])) {
         unalteredKey.add(i + "," + j);
         unalteredValue.add(postProbabilities[i][j]);
      }
      if ((postProbabilities[i][j] - preProbabilities[i][j]) > meanStandardDeviation[i]) {
         addedKey.add(i + "," + j);
         addedValue.add(postProbabilities[i][j]);
      }
      if ((preProbabilities[i][j] - postProbabilities[i][j]) > meanStandardDeviation[i]) {
         deletedKey.add(i + "," + j);
         deletedValue.add(postProbabilities[i][j]);
      }
   }
}
```

I will set for each value if it belongs to unaltered, added or deleted category. UNALTERED means that the facet_field is present both in the current choice and in the possible next choice the user could do when hover on a checkbox. ADDED means that the facet_field is present only in the possible next choice and in the current choice its value is insignificant. DELETED means that the facet_field is present only in the current choice and in the possible next choice its value is insignificant.

```
sortAlg = new sorting.algorithm.SortingAlgorithm();
```

```
sortAlg.sort(addedKey, addedValue);
addedValue = sortAlg.getData();
addedKey = sortAlg.getKey();

sortAlg.sort(deletedKey, deletedValue);
deletedValue = sortAlg.getData();
deletedKey = sortAlg.getKey();

sortAlg.sort(unalteredKey, unalteredValue);
unalteredValue = sortAlg.getData();
unalteredKey = sortAlg.getKey();
```

Sort the three new ArrayList (we have implemented an optimized solution of the quicksort algorithm).

```
System.out.print("<table>");
System.out.print("<thead>");
System.out.print("<tr><th>UNALTERED: " + unalteredKey.size() + "</th>"
      + "<th> ADDED: " + addedKey.size() + "</th><th> DELETED: " + deletedKey.size() + "</th></tr>");
System.out.print("</thead>");
System.out.print("<tbody>");
for (int i = 0; i < 5; i++) {
   if (i % 2 == 0) {
      System.out.print("<tr>");
   } else {
      System.out.print("<tr class='alternate'>");
   }
   System.out.print("<td>");
   if (unalteredKey.size() > i) {
      x = Integer.parseInt(unalteredKey.get(i).substring(0, unalteredKey.get(i).indexOf(",")));
      y = Integer.parseInt(unalteredKey.get(i).substring(unalteredKey.get(i).indexOf(",") + 1));
      System.out.println(NameField[x] + ":  " + NameValue[x][y]);
   }
   System.out.print("</td>");
   System.out.print("<td>");
   if (addedKey.size() > i) {
      x = Integer.parseInt(addedKey.get(i).substring(0, addedKey.get(i).indexOf(",")));
      y = Integer.parseInt(addedKey.get(i).substring(addedKey.get(i).indexOf(",") + 1));
      System.out.println(NameField[x] + ":  " + NameValue[x][y]);
   }
   System.out.print("</td>");
   System.out.print("<td>");
   if (deletedKey.size() > i) {
```

```
            x = Integer.parseInt(deletedKey.get(i).substring(0, deletedKey.get(i).indexOf(",")));
            y = Integer.parseInt(deletedKey.get(i).substring(deletedKey.get(i).indexOf(",") + 1));
            System.out.println(NameField[x] + ":  " + NameValue[x][y]);
          }
          System.out.print("</td>");
          System.out.print("</tr>");
        }
        System.out.print("</tbody>");
        System.out.print("</table>");
    }
```

Make the response result like an HTML table and show only the Top 5 results of each category.

8 CONCLUSIONS

Graphical models are a versatile tool that have been applied to many database problems such as selectivity estimation, sensor network data, management, information extraction, data integration to name a few. In this thesis, analyzing the Faceted Browsing case study, we presented a simple and intuitive framework for managing large-scale uncertain data using graphical models, that allows us to capture complex uncertainties and correlations in the data in a uniform manner. We showed how the problem of facets evaluation in uncertain databases visualization can be seen to be equivalent to probabilistic inference in an appropriately constructed graphical model. This equivalence enables us to employ the formidable machinery developed in the probabilistic reasoning literature over the years for

answering queries over probabilistic databases. We believe it will also lead to a deeper understanding of how to devise more efficient inference algorithms for large-scale, structured probabilistic models. The world of today populated by Big Data is fast and complex and thus its informations. Therefore, new generation information systems must be able to manage this complexity. In this thesis I have shown how a solution based on Bayesian networks represent an effective and successful approach in any kind of situation in which it is difficult to assess the relationships and dependencies between data appearing without any order. Bayesian networks have incredible power to offer assistance in a wide range of endeavors. They support the use of probabilistic inference to update and revise belief values. Bayesian networks readily permit qualitative inferences without the computational inefficiencies of traditional joint probability determinations. In doing so, they support complex inference modeling including rational decision making systems, value of information and sensitivity analysis. As such, they are useful for causality analysis and through statistical induction they support a form of automated learning. This learning can involve parametric discovery, network discovery, and causal relationship discovery.

REFERENCES

[1] S. Emmott, "Towards 2020 Science", Microsoft Research Ltd, 2006.

[2] J. C. D. Te'eni, "Human Computer Interaction: Developing Effective Organizational Information Systems", Hoboken: John Wiley & Sons, 2007.

[3] F. V. G. Riva, "Ambient Intelligence: The Evolution of Technology, Communication and Cognition towards the Future of HCI", Fairfax: IOS Press, 2005.

[4] W. W. M.T. Maybury, "Readings in Intelligent User Interfaces", San Francisco: Morgan Kaufmann Press, 1998.

[5] A. Kirlik, "Adaptive Perspectives on Human-Technology Interaction", Oxford: Oxford University Press, 2006.

[6] A. Greenfield, "Everyware: The Dawning Age of Ubiquitous Computing", New York: New Preachpit Press, 2006.

[7] N. S. A. Jaimes, "Multimodal human computer interaction: a survey", Computer Vision and Image Understanding, 2007.

[8] S. Jenson, "The Simplicity Shift: Innovative Design Tactics in a Corporate World", Cambridge, UK: Cambridge University Press, 2002.

[9] S. B. M. Johnston, "MATCHKiosk: a Multimodal Interactive City Guide", Barcelona: Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions, 2004.

[10] W. B. A. Kapoor, "Automatic prediction of frustration", International Journal of Human-Computer Studies, 2007.

[11] Z. D. C. Busso, "Analysis of emotion recognition using facial expressions, speech and multimodal information", Palo Alto, CA: Proceedings of the 6th International Conference on Multimodal Interfaces, 2004.

[12] B. University, ""Robotic Surgery: Neuro-Surgery"," [Online]. Available: http://biomed.brown.edu/Courses/BI108/BI108_2005_Groups/04/neurology.html. [Accessed 15 10 2007].

[13] SAS White Paper, "Big Data Meets Big Data Analytics", 2012.

[14] C. C. Consortium, "Advancing Discovery in Science and Engineering", 2011.

[15] C. C. Consortium, "Advancing Personalized Education", 2011.

[16] C. C. Consortium, "Smart Health and Wellbeing", 2011.

[17] McKinsey Global Institute, "Big data: The next frontier for innovation, competition, and productivity", McKinsey Global Institute, 2011.

[18] C. C. Consortium, "A Sustainable Future", 2011.

[19] H. V. J. Mark Flood, "Using Data for Systemic Financial Risk Management", Innovative Data Systems Research, 2011.

[20] G. Group, "Pattern-Based Strategy: Getting Value from Big Data", 2011.

[21] F. V. Jensen, "Bayesian Networks and Decision Graphs", NY: Springer-Verlag, 2001.

[22] R. Dechter, "Bucket Elimination: A unifying framework for probabilistic inference", Cambridge: MIT Press, 1996.

[23] M. I. Jordan, "Probabilistic inference in graphical models", Berkeley: Division of Computer Science and Department of Statistics.

[24] F. R. Kschischang, "Factor graphs and the Sum-Product Algorithm", IEEE Transactions on Information Theory, 2001.

[25] R. J. McEliece, "Turbo decoding as an instance of Pearl's belief propagation algorithm", IEEE Journal on Selected Areas in Communication, 1996.

[26] R. Gilks W., "Markov Chain Monte Carlo in Practice: Interdisciplinary Statistics", Chapman & Hall, 1995.

[27] A. Darwiche, "What are Bayesian networks and why are their applications growing across all fields?".

[28] A. Darwiche, "Modeling and Reasoning with Bayesian Networks", UCLA, 2009.

[29] A. Darwiche, "A differential approach to inference in Bayesian Networks", Journal of ACM, 2003.

[30] F. Koller D., "Probabilistic Graphical Models: Principles and Techniques", Cambridge: MIT Press, 2009.

[31] C. G. P. Spirtes, "An algorithm for fast recovery of sparse causal graphs", Social Science Computer, 1991.

[32] C. G. P. Spirtes, "Causation, Prediction and Search", Cambridge: MIT Press, 2000.

[33] E. H. G. Cooper, "A Bayesian method for constructing Bayesian belief networks from databases", UAI '91, 1991.

[34] F. J. D. M. Arias, "Carmen: An open source project for probabilistic graphical models", PGM'08, 2008.

[35] T. E. Consortium, "Elvira: An environment for creating and using probabilistic graphical models", PGM'02, 2002.

[36] F. J. D. M. Arias, "ProbModelXML. A format for encoding probabilistic graphical models", Madrid, 2011.

[37] H. S. I. Beinlich, "The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks", London, 1989.

[38] G. C. C. Glymour, "Computation, causation and discovery", Cambridge: MIT Press, 1999.

[39] F. D. M. Druzdzel, "Combining knowledge from different sources in probabilistic models", Journal of Machine Learning Research, 2003.

[40] E. S. R. Teach, "An analysis of physician's attitudes", MYCIN Experiments of the Stanford Heuristic Programming Project, 1984.

[41] G. M. D. Tunkelang, "Faceted Search", Morgan and Claypool Publishers, 2009.

[42] P. S. M. Hearst, "Faceted metadata for information architecture and search", CHI Tutorial, 2006.

[43] V. H. A. Kashyap, "Facetor: cost-driven exploration of faceted query results", CIKM, , 2010.

[44] J. R. D. Dash, "Dynamic faceted search for discovery-driven analysis", CIKM, , 2008.

[45] G. Nudelman, "Designing search: UX strategies for ecommerce success", Wiley, 2011.

[46] M. A. Hearst, "Clustering versus faceted categories for information exploration", ACM, 2006.

[47] M. G. A. Deshpande, "Independence is good: dependency-based histogram synopses for high-dimensional data", ACM, 2001.

[48] P. H. P. Brown, "Automatic discovery of fuzzy algebraic constraints in relational data", VLDB Endowment, 2003.