

Bater Makhabel, Pradeepta Mishra,
Nathan Danneman, Richard Heimann

R: Mining Spatial, Text, Web, and Social Media Data

Learning Path

Create and customize data mining algorithms



Packt>

R: Mining Spatial, Text, Web, and Social Media Data

Create and customize data mining algorithms

A course in three modules



BIRMINGHAM - MUMBAI

R: Mining Spatial, Text, Web, and Social Media Data

Copyright © 2016 Packt Publishing

All rights reserved. No part of this course may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this course to ensure the accuracy of the information presented. However, the information contained in this course is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this course.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this course by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Published on: April 2017

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78829-374-7

www.packtpub.com

Credits

Authors

Bater Makhabel

Pradeepta Mishra

Nathan Danneman

Richard Heimann

Reviewers

Jason H.D. Cho

Gururaghav Gopal

Vibhav Kamath

Hasan Kurban

Alexey Grigorev

Carlos J. Gil Bellosta

Vibhav Vivek Kamath

Feng Mai

Ajay Ohri

Yanchang Zhao

Content Development Editor

Trusha Shriyan

Graphics

Jason Monterio

Production Coordinator

Melwyn Dsa

Preface

The necessity to handle many, complex statistical analysis projects is hitting statisticians and analysts across the globe. With increasing interest in data analysis, R offers a free and open source environment that is perfect for both learning and deploying predictive modeling solutions in the real world. With its constantly growing community and plethora of packages, R offers functionality for dealing with a truly vast array of problems.

It's been decades since the R programming language was born and has become an eminent and well known not only within the community of scientist, but also in the wider community of developers. It has grown into a powerful tool to help developers produce efficient and consistent source code for data related tasks. The R development team and independent contributors have created good documentation so getting started programming with R isn't that hard.

What this learning path covers

Module 1, Learning Data mining with R, will teach you how to manipulate data with R using code snippets and be introduced to mining frequent patterns, association, and correlations while working with R programs. You will discover how to write code for various predication models, stream data, and time-series data. You will also be introduced to solutions written in R based on RHadoop projects. You will finish this module by feeling confident in your ability to know which data mining algorithm to apply in any situation.

Module 2, R Data Mining Blueprints, explores data mining techniques and shows you how to apply different mining concepts to various statistical and data applications in a wide range of fields. You will learn about R and its application to data mining, and give you relevant and useful information you can use to develop and improve your applications. This module will help you complete complex data mining cases and guide you through handling issues you might encounter during projects.

Module 3, Social Media Mining with R, begins by introducing you to the topic of social media data, including its sources and properties. It then explains the basics of R programming in a straightforward, unassuming way. Thereafter, you will be made aware of the inferential dangers associated with social media data and how to avoid them, before describing and implementing a suite of social media mining techniques.

What you need for this learning path

Any modern PC with installed Windows, Linux or Mac OS should be sufficient to run the code samples in the book. All the software used in the book is open source and freely available on the Web:

<http://www.r-project.org/>

Who this learning path is for

This learning path is for R developers who are looking up to making a career in data analysis or data mining. Those who come across data mining problems of different complexities from web, text, numerical, political, and social media domains will find all information in this single learning path.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this course – what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail feedback@packtpub.com, and mention the course's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt course, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for this course from your account at <http://www.packtpub.com>. If you purchased this course elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

You can download the code files by following these steps:

1. Log in or register to our website using your e-mail address and password.
2. Hover the mouse pointer on the **SUPPORT** tab at the top.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the course in the **Search** box.
5. Select the course for which you're looking to download the code files.
6. Choose from the drop-down menu where you purchased this course from.
7. Click on **Code Download**.

You can also download the code files by clicking on the **Code Files** button on the course's webpage at the Packt Publishing website. This page can be accessed by entering the course's name in the **Search** box. Please note that you need to be logged in to your Packt account.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR / 7-Zip for Windows
- Zipeg / iZip / UnRarX for Mac
- 7-Zip / PeaZip for Linux

The code bundle for the course is also hosted on GitHub at <https://github.com/PacktPublishing/R-Mining-spatial-text-web-and-social-media-data/>. We also have other code bundles from our rich catalog of books, videos, and courses available at <https://github.com/PacktPublishing/>. Check them out!

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our courses—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this course. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your course, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to <https://www.packtpub.com/books/content/support> and enter the name of the course in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this course, you can contact us at questions@packtpub.com, and we will do our best to address the problem.

Module 1: Learning Data mining with R

| | |
|--|-----------|
| Chapter 1: Warming Up | 1 |
| Big data | 2 |
| Data source | 4 |
| Data mining | 4 |
| Social network mining | 8 |
| Text mining | 11 |
| Web data mining | 12 |
| Why R? | 14 |
| Statistics | 14 |
| Machine learning | 16 |
| Data attributes and description | 17 |
| Data cleaning | 21 |
| Data integration | 23 |
| Data dimension reduction | 23 |
| Data transformation and discretization | 25 |
| Visualization of results | 27 |
| Time for action | 28 |
| Summary | 29 |
| Chapter 2: Mining Frequent Patterns, Associations, and Correlations | 31 |
| An overview of associations and patterns | 32 |
| Market basket analysis | 38 |
| Hybrid association rules mining | 58 |
| Mining sequence dataset | 59 |
| The R implementation | 62 |
| High-performance algorithms | 65 |
| Time for action | 65 |
| Summary | 66 |

| | |
|--|------------|
| Chapter 3: Classification | 67 |
| Classification | 68 |
| Generic decision tree induction | 70 |
| High-value credit card customers classification using ID3 | 76 |
| Web spam detection using C4.5 | 82 |
| Web key resource page judgment using CART | 90 |
| Trojan traffic identification method and Bayes classification | 93 |
| Identify spam e-mail and Naïve Bayes classification | 98 |
| Rule-based classification of player types in computer games and rule-based classification | 102 |
| Time for action | 108 |
| Summary | 109 |
| Chapter 4: Advanced Classification | 111 |
| Ensemble (EM) methods | 111 |
| Biological traits and the Bayesian belief network | 118 |
| Protein classification and the k-Nearest Neighbors algorithm | 120 |
| Document retrieval and Support Vector Machine | 121 |
| Classification using frequent patterns | 128 |
| Classification using the backpropagation algorithm | 131 |
| Time for action | 137 |
| Summary | 137 |
| Chapter 5: Cluster Analysis | 139 |
| Search engines and the k-means algorithm | 142 |
| Automatic abstraction of document texts and the k-medoids algorithm | 150 |
| The CLARA algorithm | 153 |
| CLARANS | 155 |
| Unsupervised image categorization and affinity propagation clustering | 156 |
| News categorization and hierarchical clustering | 160 |
| Time for action | 166 |
| Summary | 166 |
| Chapter 6: Advanced Cluster Analysis | 169 |
| Customer categorization analysis of e-commerce and DBSCAN | 169 |
| Clustering web pages and OPTICS | 172 |
| Visitor analysis in the browser cache and DENCLUE | 175 |
| Recommendation system and STING | 180 |
| Web sentiment analysis and CLIQUE | 181 |
| Opinion mining and WAVE clustering | 183 |
| User search intent and the EM algorithm | 186 |
| Customer purchase data analysis and clustering high-dimensional data | 188 |
| SNS and clustering graph and network data | 191 |

| | |
|--|------------|
| Time for action | 193 |
| Summary | 193 |
| Chapter 7: Outlier Detection | 195 |
| Credit card fraud detection and statistical methods | 196 |
| Activity monitoring – the detection of fraud involving mobile phones and proximity-based methods | 199 |
| Intrusion detection and density-based methods | 205 |
| Intrusion detection and clustering-based methods | 210 |
| Monitoring the performance of the web server and classification-based methods | 212 |
| Detecting novelty in text, topic detection, and mining contextual outliers | 214 |
| Collective outliers on spatial data | 217 |
| Outlier detection in high-dimensional data | 219 |
| Time for action | 221 |
| Summary | 221 |
| Chapter 8: Mining Stream, Time-series, and Sequence Data | 223 |
| The credit card transaction flow and STREAM algorithm | 224 |
| Predicting future prices and time-series analysis | 227 |
| Stock market data and time-series clustering and classification | 230 |
| Web click streams and mining symbolic sequences | 233 |
| Mining sequence patterns in transactional databases | 237 |
| Time for action | 238 |
| Summary | 239 |
| Chapter 9: Graph Mining and Network Analysis | 241 |
| Graph mining | 241 |
| Mining frequent subgraph patterns | 242 |
| Social network mining | 246 |
| Time for action | 249 |
| Summary | 249 |
| Chapter 10: Mining Text and Web Data | 251 |
| Text mining and TM packages | 252 |
| Text summarization | 252 |
| The question answering system | 257 |
| Genre categorization of web pages | 258 |
| Categorizing newspaper articles and newswires into topics | 259 |
| Web usage mining with web logs | 262 |
| Time for action | 265 |
| Summary | 265 |
| Appendix: Algorithms and Data Structures | 267 |

Module 1

Learning Data mining with R

Develop key skills and techniques with R to create and customize data mining algorithms

1

Warming Up

In this chapter, you will learn basic data mining terms such as data definition, preprocessing, and so on.

The most important data mining algorithms will be illustrated with R to help you grasp the principles quickly, including but not limited to, classification, clustering, and outlier detection. Before diving right into data mining, let's have a look at the topics we'll cover:

- Data mining
- Social network mining
- Text mining
- Web data mining
- Why R
- Statistics
- Machine learning
- Data attributes and description
- Data measuring
- Data cleaning
- Data integration
- Data reduction
- Data transformation and discretization
- Visualization of results

In the history of humankind, the results of data from every aspect is extensive, for example websites, social networks by user's e-mail or name or account, search terms, locations on map, companies, IP addresses, books, films, music, and products.

Data mining techniques can be applied to any kind of old or emerging data; each data type can be best dealt with using certain, but not all, techniques. In other words, the data mining techniques are constrained by data type, size of the dataset, context of the tasks applied, and so on. Every dataset has its own appropriate data mining solutions.

New data mining techniques always need to be researched along with new data types once the old techniques cannot be applied to it or if the new data type cannot be transformed onto the traditional data types. The evolution of stream mining algorithms applied to Twitter's huge source set is one typical example. The graph mining algorithms developed for social networks is another example.

The most popular and basic forms of data are from databases, data warehouses, ordered/sequence data, graph data, text data, and so on. In other words, they are federated data, high dimensional data, longitudinal data, streaming data, web data, numeric, categorical, or text data.

Big data

Big data is large amount of data that does not fit in the memory of a single machine. In other words, the size of data itself becomes a part of the issue when studying it. Besides volume, two other major characteristics of big data are variety and velocity; these are the famous three Vs of big data. Velocity means data process rate or how fast the data is being processed. Variety denotes various data source types. Noises arise more frequently in big data source sets and affect the mining results, which require efficient data preprocessing algorithms.

As a result, distributed filesystems are used as tools for successful implementation of parallel algorithms on large amounts of data; it is a certainty that we will get even more data with each passing second. Data analytics and visualization techniques are the primary factors of the data mining tasks related to massive data. The characteristics of massive data appeal to many new data mining technique-related platforms, one of which is RHadoop. We'll be describing this in a later section.

Some data types that are important to big data are as follows:

- The data from the camera video, which includes more metadata for analysis to expedite crime investigations, enhanced retail analysis, military intelligence, and so on.
- The second data type is from embedded sensors, such as medical sensors, to monitor any potential outbreaks of virus.

- The third data type is from entertainment, information freely published through social media by anyone.
- The last data type is consumer images, aggregated from social medias, and tagging on these like images are important.

Here is a table illustrating the history of data size growth. It shows that information will be more than double every two years, changing the way researchers or companies manage and extract value through data mining techniques from data, revealing new data mining studies.

| Year | Data Sizes | Comments |
|-----------|------------|---|
| N/A | | 1 MB (Megabyte) = 2^{20} . The human brain holds about 200 MB of information. |
| N/A | | 1 PB (Petabyte) = 2^{50} . It is similar to the size of 3 years' observation data for Earth by NASA and is equivalent of 70.8 times the books in America's Library of Congress. |
| 1999 | 1 EB | 1 EB (Exabyte) = 2^{60} . The world produced 1.5 EB of unique information. |
| 2007 | 281 EB | The world produced about 281 Exabyte of unique information. |
| 2011 | 1.8 ZB | 1 ZB (Zetabyte)= 2^{70} . This is all data gathered by human beings in 2011. |
| Very soon | | 1 YB(Yottabytes)= 2^{80} . |

Scalability and efficiency

Efficiency, scalability, performance, optimization, and the ability to perform in real time are important issues for almost any algorithms, and it is the same for data mining. There are always necessary metrics or benchmark factors of data mining algorithms.

As the amount of data continues to grow, keeping data mining algorithms effective and scalable is necessary to effectively extract information from massive datasets in many data repositories or data streams.

The storage of data from a single machine to wide distribution, the huge size of many datasets, and the computational complexity of the data mining methods are all factors that drive the development of parallel and distributed data-intensive mining algorithms.

Data source

Data serves as the input for the data mining system and data repositories are important. In an enterprise environment, database and logfiles are common sources. In web data mining, web pages are the source of data. The data that continuously fetched various sensors are also a typical data source.

Here are some free online data sources particularly helpful to learn about data mining:

- **Frequent Itemset Mining Dataset Repository:** A repository with datasets for methods to find frequent itemsets (<http://fimi.ua.ac.be/data/>).
- **UCI Machine Learning Repository:** This is a collection of dataset, suitable for classification tasks (<http://archive.ics.uci.edu/ml/>).
- **The Data and Story Library at statlib:** DASL (pronounced "dazzle") is an online library of data files and stories that illustrate the use of basic statistics methods. We hope to provide data from a wide variety of topics so that statistics teachers can find real-world examples that will be interesting to their students. Use DASL's powerful search engine to locate the story or data file of interest. (<http://lib.stat.cmu.edu/DASL/>)
- **WordNet:** This is a lexical database for English (<http://wordnet.princeton.edu>)



Data mining

Data mining is the discovery of a *model* in data; it's also called exploratory data analysis, and discovers useful, valid, unexpected, and understandable knowledge from the data. Some goals are shared with other sciences, such as statistics, artificial intelligence, machine learning, and pattern recognition. Data mining has been frequently treated as an algorithmic problem in most cases. Clustering, classification, association rule learning, anomaly detection, regression, and summarization are all part of the tasks belonging to data mining.

The data mining methods can be summarized into two main categories of data mining problems: feature extraction and summarization.

Feature extraction

This is to extract the most prominent features of the data and ignore the rest. Here are some examples:

- **Frequent itemsets:** This model makes sense for data that consists of *baskets* of small sets of items.
- **Similar items:** Sometimes your data looks like a collection of sets and the objective is to find pairs of sets that have a relatively large fraction of their elements in common. It's a fundamental problem of data mining.

Summarization

The target is to summarize the dataset succinctly and approximately, such as clustering, which is the process of examining a collection of *points* (data) and grouping the points into *clusters* according to some measure. The goal is that points in the same cluster have a small distance from one another, while points in different clusters are at a large distance from one another.

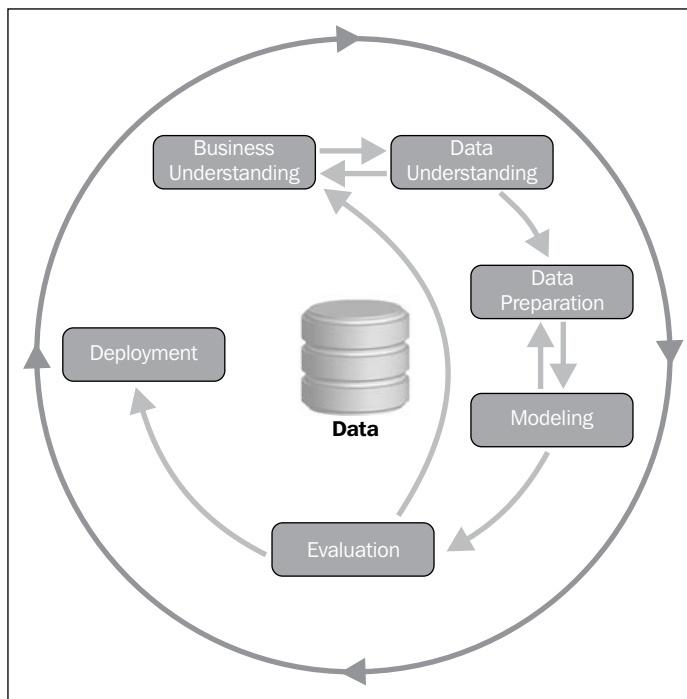
The data mining process

There are two popular processes to define the data mining process in different perspectives, and the more widely adopted one is CRISP-DM:

- **Cross-Industry Standard Process for Data Mining (CRISP-DM)**
- **Sample, Explore, Modify, Model, Assess (SEMMA)**, which was developed by the SAS Institute, USA

CRISP-DM

There are six phases in this process that are shown in the following figure; it is not rigid, but often has a great deal of backtracking:



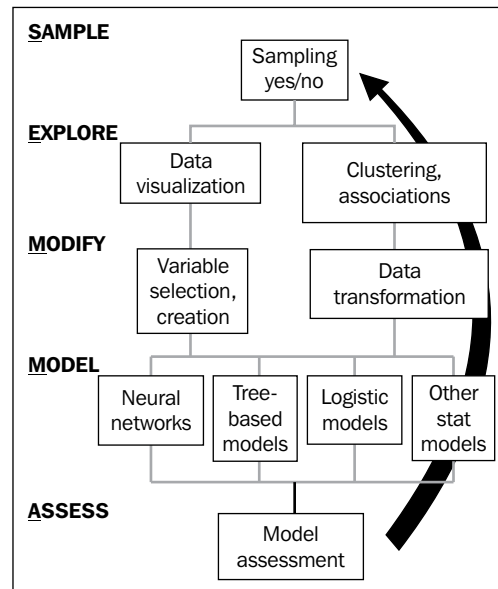
Let's look at the phases in detail:

- **Business understanding:** This task includes determining business objectives, assessing the current situation, establishing data mining goals, and developing a plan.
- **Data understanding:** This task evaluates data requirements and includes initial data collection, data description, data exploration, and the verification of data quality.
- **Data preparation:** Once available, data resources are identified in the last step. Then, the data needs to be selected, cleaned, and then built into the desired form and format.

- **Modeling:** Visualization and cluster analysis are useful for initial analysis. The initial association rules can be developed by applying tools such as generalized rule induction. This is a data mining technique to discover knowledge represented as rules to illustrate the data in the view of causal relationship between conditional factors and a given decision/outcome. The models appropriate to the data type can also be applied.
- **Evaluation :**The results should be evaluated in the context specified by the business objectives in the first step. This leads to the identification of new needs and in turn reverts to the prior phases in most cases.
- **Deployment:** Data mining can be used to both verify previously held hypotheses or for knowledge.

SEMMA

Here is an overview of the process for SEMMA:



Let's look at these processes in detail:

- **Sample:** In this step, a portion of a large dataset is extracted
- **Explore:** To gain a better understanding of the dataset, unanticipated trends and anomalies are searched in this step
- **Modify:** The variables are created, selected, and transformed to focus on the model construction process

- **Model:** A variable combination of models is searched to predict a desired outcome
- **Assess:** The findings from the data mining process are evaluated by its usefulness and reliability

Social network mining

As we mentioned before, data mining finds a *model* on data and the mining of social network finds the *model* on graph data in which the social network is represented.

Social network mining is one application of web data mining; the popular applications are social sciences and bibliometry, PageRank and HITS, shortcomings of the coarse-grained graph model, enhanced models and techniques, evaluation of topic distillation, and measuring and modeling the Web.

Social network

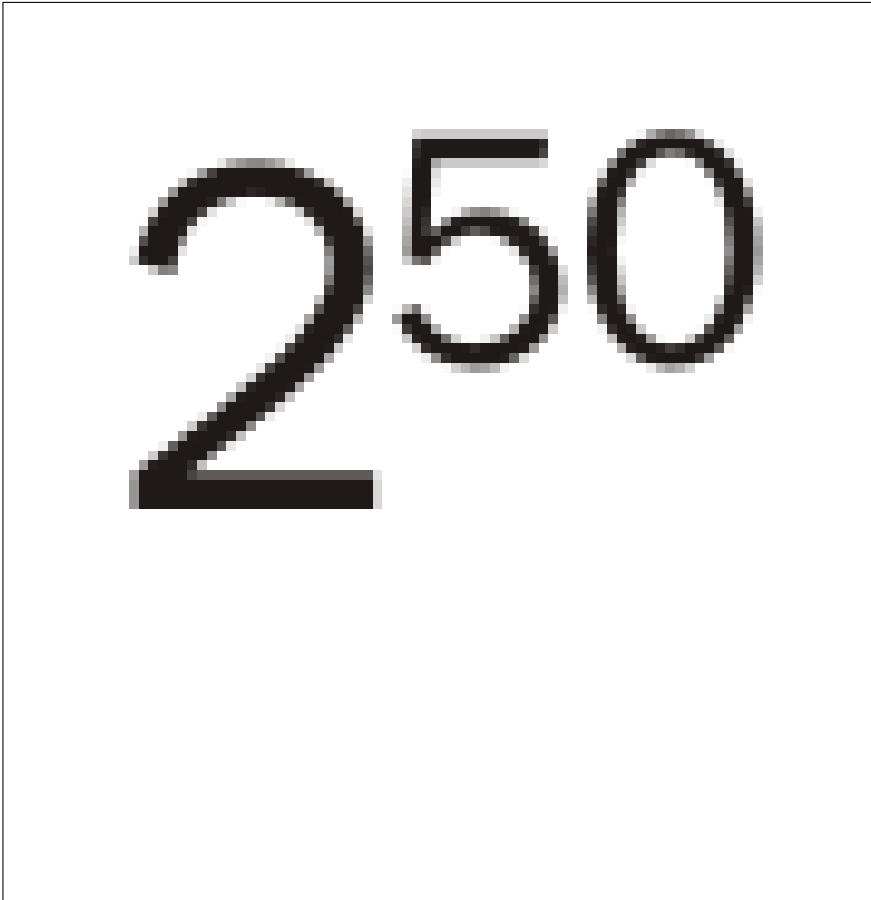
When it comes to the discussion of social networks, you will think of Facebook, Google+, LinkedIn, and so on. The essential characteristics of a social network are as follows:

- There is a collection of entities that participate in the network. Typically, these entities are people, but they could be something else entirely.
- There is at least one relationship between the entities of the network. On Facebook, this relationship is called friends. Sometimes, the relationship is all-or-nothing; two people are either friends or they are not. However, in other examples of social networks, the relationship has a degree. This degree could be discrete, for example, friends, family, acquaintances, or none as in Google+. It could be a real number; an example would be the fraction of the average day that two people spend talking to each other.
- There is an assumption of nonrandomness or locality. This condition is the hardest to formalize, but the intuition is that relationships tend to cluster. That is, if entity *A* is related to both *B* and *C*, then there is a higher probability than average that *B* and *C* are related.

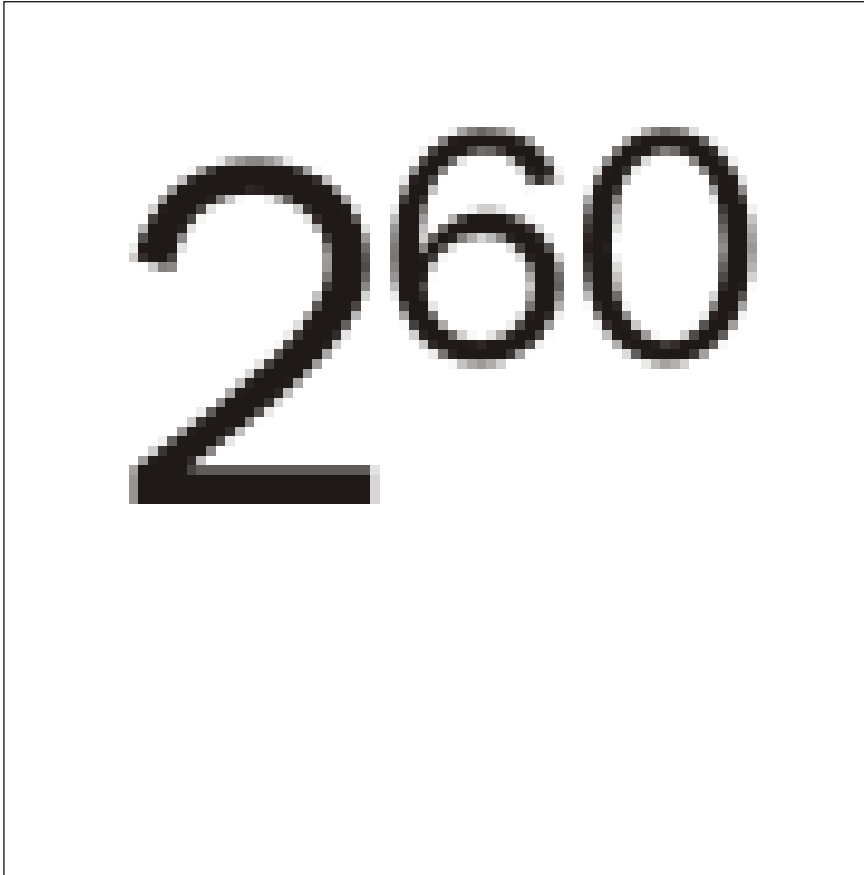
Here are some varieties of social networks:

- **Telephone networks:** The nodes in this network are phone numbers and represent individuals
- **E-mail networks:** The nodes represent e-mail addresses, which represent individuals
- **Collaboration networks:** The nodes here represent individuals who published research papers; the edge connecting two nodes represent two individuals who published one or more papers jointly

Social networks are modeled as undirected graphs. The entities are the nodes, and an edge connects two nodes if the nodes are related by the relationship that characterizes the network. If there is a degree associated with the relationship, this degree is represented by labeling the edges.



Here is an example in which Coleman's High School Friendship Data from the `sna` R package is used for analysis. The data is from a research on friendship ties between 73 boys in a high school in one chosen academic year; reported ties for all informants are provided for two time points (fall and spring). The dataset's name is `coleman`, which is an array type in R language. The node denotes a specific student and the line represents the tie between two students.



Text mining

Text mining is based on the data of text, concerned with exacting relevant information from large natural language text, and searching for interesting relationships, syntactical correlation, or semantic association between the extracted entities or terms. It is also defined as automatic or semiautomatic processing of text. The related algorithms include text clustering, text classification, natural language processing, and web mining.

One of the characteristics of text mining is text mixed with numbers, or in other point of view, the hybrid data type contained in the source dataset. The text is usually a collection of unstructured documents, which will be preprocessed and transformed into a numerical and structured representation. After the transformation, most of the data mining algorithms can be applied with good effects.

The process of text mining is described as follows:

- Text mining starts from preparing the text corpus, which are reports, letters and so forth
- The second step is to build a semistructured text database that is based on the text corpus
- The third step is to build a term-document matrix in which the term frequency is included
- The final result is further analysis, such as text analysis, semantic analysis, information retrieval, and information summarization

Information retrieval and text mining

Information retrieval is to help users find information, most commonly associated with online documents. It focuses on the acquisition, organization, storage, retrieval, and distribution for information. The task of **Information Retrieval (IR)** is to retrieve relevant documents in response to a query. The fundamental technique of IR is measuring similarity. Key steps in IR are as follows:

- Specify a query. The following are some of the types of queries:
 - **Keyword query:** This is expressed by a list of keywords to find documents that contain at least one keyword
 - **Boolean query:** This is constructed with Boolean operators and keywords
 - **Phrase query:** This is a query that consists of a sequence of words that makes up a phrase

- **Proximity query:** This is a downgrade version of the phrase queries and can be a combination of keywords and phrases
 - **Full document query:** This query is a full document to find other documents similar to the query document
 - **Natural language questions:** This query helps to express users' requirements as a natural language question
- Search the document collection.
 - Return the subset of relevant documents.

Mining text for prediction

Prediction of results from text is just as ambitious as predicting numerical data mining and has similar problems associated with numerical classification. It is generally a classification issue.

Prediction from text needs prior experience, from the sample, to learn how to draw a prediction on new documents. Once text is transformed into numeric data, prediction methods can be applied.

Web data mining

Web mining aims to discover useful information or knowledge from the web hyperlink structure, page, and usage data. The Web is one of the biggest data sources to serve as the input for data mining applications.

Web data mining is based on IR, **machine learning (ML)**, statistics, pattern recognition, and data mining. Web mining is not purely a data mining problem because of the heterogeneous and semistructured or unstructured web data, although many data mining approaches can be applied to it.

Web mining tasks can be defined into at least three types:

- **Web structure mining:** This helps to find useful information or valuable structural summary about sites and pages from hyperlinks
- **Web content mining:** This helps to mine useful information from web page contents
- **Web usage mining:** This helps to discover user access patterns from web logs to detect intrusion, fraud, and attempted break-in

The algorithms applied to web data mining are originated from classical data mining algorithms. They share many similarities, such as the mining process; however, differences exist too. The characteristics of web data mining makes it different from data mining for the following reasons:

- The data is unstructured
- The information of the Web keeps changing and the amount of data keeps growing
- Any data type is available on the Web, such as structured and unstructured data
- Heterogeneous information is on the web; redundant pages are present too
- Vast amounts of information on the web is linked
- The data is noisy

Web data mining differentiates from data mining by the huge dynamic volume of source dataset, a big variety of data format, and so on. The most popular data mining tasks related to the Web are as follows:

- **Information extraction (IE):** The task of IE consists of a couple of steps, tokenization, sentence segmentation, part-of-speech assignment, named entity identification, phrasal parsing, sentential parsing, semantic interpretation, discourse interpretation, template filling, and merging.
- **Natural language processing (NLP):** This researches the linguistic characteristics of human-human and human-machine interactive, models of linguistic competence and performance, frameworks to implement process with such models, processes'/models' iterative refinement, and evaluation techniques for the result systems. Classical NLP tasks related to web data mining are tagging, knowledge representation, ontologies, and so on.
- **Question answering:** The goal is to find the answer from a collection of text to questions in natural language format. It can be categorized into slot filling, limited domain, and open domain with bigger difficulties for the latter. One simple example is based on a predefined FAQ to answer queries from customers.
- **Resource discovery:** The popular applications are collecting important pages preferentially; similarity search using link topology, topical locality and focused crawling; and discovering communities.

Why R?

R is a high-quality, cross-platform, flexible, widely used open source, free language for statistics, graphics, mathematics, and data science – created by statisticians for statisticians.

R contains more than 5,000 algorithms and millions of users with domain knowledge worldwide, and it is supported by a vibrant and talented community of contributors. It allows access to both well-established and experimental statistical techniques.

R is a free, open source software environment maintained by R-projects for statistical computing and graphics, and the R source code is available under the terms of the Free Software Foundation's GNU General Public License. R compiles and runs on a wide variety for a variety of platforms, such as UNIX, LINUX, Windows, and Mac OS.

What are the disadvantages of R?

There are three shortages of R:

- One is that it is memory bound, so it requires the entire dataset store in memory (RAM) to achieve high performance, which is also called in-memory analytics.
- Similar to other open source systems, anyone can create and contribute package with strict or less testing. In other words, packages contributing to R communities are bug-prone and need more testing to ensure the quality of codes.
- R seems slow than some other commercial languages.

Fortunately, there are packages available to overcome these problems. There are some solutions that can be categorized as parallelism solutions; the essence here is to spread work across multiple CPUs that overcome the R shortages that were just listed. Good examples include, but are not limited to, RHadoop. You will read more on this topic soon in the following sections. You can download the SNOW add-on package and the Parallel add-on package from **Comprehensive R Archive Network (CRAN)**.

Statistics

Statistics studies the collection, analysis, interpretation or explanation, and presentation of data. It serves as the foundation of data mining and the relations will be illustrated in the following sections.

Statistics and data mining

Statisticians were the first to use the term data mining. Originally, data mining was a derogatory term referring to attempts to extract information that was not supported by the data. To some extent, data mining constructs statistical models, which is an underlying distribution, used to visualize data.

Data mining has an inherent relationship with statistics; one of the mathematical foundations of data mining is statistics, and many statistics models are used in data mining.

Statistical methods can be used to summarize a collection of data and can also be used to verify data mining results.

Statistics and machine learning

Along with the development of statistics and machine learning, there is a continuum between these two subjects. Statistical tests are used to validate the machine learning models and to evaluate machine learning algorithms. Machine learning techniques are incorporated with standard statistical techniques.

Statistics and R

R is a statistical programming language. It provides a huge amount of statistical functions, which are based on the knowledge of statistics. Many R add-on package contributors come from the field of statistics and use R in their research.

The limitations of statistics on data mining

During the evolution of data mining technologies, due to statistical limits on data mining, one can make errors by trying to extract what really isn't in the data.

Bonferroni's Principle is a statistical theorem otherwise known as **Bonferroni correction**. You can assume that big portions of the items you find are bogus, that is, the items returned by the algorithms dramatically exceed what is assumed.

Machine learning

The data to which a ML algorithm is applied is called a training set, which consists of a set of pairs (x, y) , called training examples. The pairs are explained as follows:

- x : This is a vector of values, often called the feature vector. Each value, or feature, can be categorical (values are taken from a set of discrete values, such as $\{S, M, L\}$) or numerical.
- y : This is the label, the classification or regression values for x .

The objective of the ML process is to discover a function $y = f(x)$ that best predicts the value of y associated with each value of x . The type of y is in principle arbitrary, but there are several common and important cases.

- y : This is a real number. The ML problem is called regression.
- y : This is a Boolean value true or false, more commonly written as +1 and -1, respectively. In this class, the problem is binary classification.
- y : Here this is a member of some finite set. The member of this set can be thought of as *classes*, and each member represents one class. The problem is multiclass classification.
- y : This is a member of some potentially infinite set, for example, a parse tree for x , which is interpreted as a sentence.

Until now, machine learning has not proved successful in situations where we can describe the goals of the mining more directly. Machine learning and data mining are two different topics, although some algorithms are shared between them – algorithms are shared especially when the goal is to extract information. There are situations where machine learning makes sense. The typical one is when we have idea of what we looking for in the dataset.

Approaches to machine learning

The major classes of algorithms are listed here. Each is distinguished by the function f .

- **Decision tree:** This form of f is a tree and each node of the tree has a function of x that determines which child or children the search must proceed for.
- **Perceptron:** These are threshold functions applied to the components of the vector $x = \{x_1, x_2, \dots, x_n\}$. A weight w_i is associated with the i th components, for each $i = 1, 2, \dots, n$, and there is a threshold $\sum_{i=1}^n w_i x_i \geq \theta$. The output is +1 if and the output is -1 otherwise.

- **Neural nets:** These are acyclic networks of perceptions, with the outputs of some perceptions used as inputs to others.
- **Instance-based learning:** This uses the entire training set to represent the function f .
- **Support-vector machines:** The result of this class is a classifier that tends to be more accurate on unseen data. The target for class separation denotes as looking for the optimal hyper-plane separating two classes by maximizing the margin between the classes' closest points.

Machine learning architecture

The data aspects of machine learning here means the way data is handled and the way it is used to build the model.

- **Training and testing:** Assuming all the data is suitable for training, separate out a small fraction of the available data as the test set; use the remaining data to build a suitable model or classifier.
- **Batch versus online learning:** The entire training set is available at the beginning of the process for batch mode; the other one is online learning, where the training set arrives in a stream and cannot be revisited after it is processed.
- **Feature selection:** This helps to figure out what features to use as input to the learning algorithm.
- **Creating a training set:** This helps to create the label information that turns data into a training set by hand.

Data attributes and description

An **attribute** is a field representing a certain feature, characteristic, or dimensions of a data object.

In most situations, data can be modeled or represented with a matrix, columns for data attributes, and rows for certain data records in the dataset. For other cases, that data cannot be represented with matrices, such as text, time series, images, audio, video, and so forth. The data can be transformed into a matrix by appropriate methods, such as feature extraction.

The type of data attributes arises from its contexts or domains or semantics, and there are numerical, non-numerical, categorical data types or text data. Two views applied to data attributes and descriptions are widely used in data mining and R. They are as follows:

- **Data in algebraic or geometric view:** The entire dataset can be modeled into a matrix; linear algebraic and abstract algebra plays an important role here.
- **Data in probability view:** The observed data is treated as multidimensional random variables; each numeric attribute is a random variable. The dimension is the data dimension. Irrespective of whether the value is discrete or continuous, the probability theory can be applied here.

To help you learn R more naturally, we shall adopt a geometric, algebraic, and probabilistic view of the data.

Here is a matrix example. The number of columns is determined by m , which is the dimensionality of data. The number of rows is determined by n , which is the size of dataset.

$$A = \begin{pmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = (X_1 \cdots X_m)$$

Where x_i denotes the i row, which is an m -tuple as follows:

$$x_i = (x_{i1}, \cdots, x_{im})$$

And x_j denotes the j column, which is an n -tuple as follows:

$$X_j = (X_{1j}, \cdots, X_{nj})$$

Numeric attributes

Numerical data is convenient to deal with because it is quantitative and allows arbitrary calculations. The properties of numerical data are the same as integer or float data.

Numeric attributes taken from a finite or countable infinite set of values are called **discrete**, for example a human being's age, which is the integer value starting from 1,150. Other attributes taken from any real values are called **continuous**. There are two main kinds of numeric types:

- **Interval-scaled:** This is the quantitative value, measured on a scale of equal unit, such as the weight of some certain fish in the scale of international metric, such as gram or kilogram.
- **Ratio-scaled:** This value can be computed by ratios between values in addition to differences between values. It is a numeric attribute with an inherent zero-point; hence, we can say a value is a multiple of another value.

Categorical attributes

The values of categorical attributes come from a set-valued domain composed of a set of symbols, such as the size of human costumes that are categorized as $\{S, M, L\}$. The categorical attributes can be divided into two groups or types:

- **Nominal:** The values in this set are unordered and are not quantitative; only the equality operation makes sense here.
- **Ordinal:** In contrast to the nominal type, the data has an ordered meaning here. The inequality operation is available here in addition to the equality operation.

Data description

The basic description can be used to identify features of data, distinguish noise, or outliers. A couple of basic statistical descriptions are as follows:

- **Measures of central tendency:** This measures the location of middle or center of a data distribution: the mean, median, mode, midrange, and so on.
- **Measure of the dispersion of the data:** This is the range, quartiles, interquartile range, and so on.

Data measuring

Data measuring is used in clustering, outlier detection, and classification. It refers to measures of proximity, similarity, and dissimilarity. The similarity value, a real value, between two tuples or data records ranges from 0 to 1, the higher the value the greater the similarity between tuples. Dissimilarity works in the opposite way; the higher the dissimilarity value, the more dissimilar are the two tuples.

For a dataset, data matrix stores the n data tuples in $n \times m$ matrix (n tuples and m attributes):

$$\begin{pmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix}$$

The dissimilarity matrix stores a collection of proximities available for all n tuples in the dataset, often in a $n \times n$ matrix. In the following matrix, $d(i, j)$ means the dissimilarity between two tuples; value 0 for highly similar or near between each other, 1 for completely same, the higher the value, the more dissimilar it is.

$$\begin{pmatrix} 0 \\ d(2,1)0 \\ d(3,1)d(3,2)0 \\ \vdots \\ d(n,1)d(n,2)\dots 0 \end{pmatrix}$$

Most of the time, the dissimilarity and similarity are related concepts. The similarity measure can often be defined using a function; the expression constructed with measures of dissimilarity, and vice versa.

Here is a table with a list of some of the most used measures for different attribute value types:

| Attribute value types | Dissimilarity |
|-----------------------|--|
| Nominal attributes | The dissimilarity between two tuples can be computed by the following equation: $d(i, j) = (p-m)/p$ Where, p is the dimension of data and m is the number of matches that is in same state. |
| Ordinal attributes | The treatment of ordinal attributes is similar to that of numeric attributes, but it needs a transformation first before applying the methods. |
| Interval-scaled | Euclidean , Manhattan , and Minkowski distances are used to calculate the dissimilarity of data tuples. |

Data cleaning

Data cleaning is one part of data quality. The aim of **Data Quality (DQ)** is to have the following:

- Accuracy (data is recorded correctly)
- Completeness (all relevant data is recorded)
- Uniqueness (no duplicated data record)
- Timeliness (the data is not old)
- Consistency (the data is coherent)

Data cleaning attempts to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data. Data cleaning is usually an iterative two-step process consisting of discrepancy detection and data transformation.

The process of data mining contains two steps in most situations. They are as follows:

- The first step is to perform audition on the source dataset to find the discrepancy.
- The second step is to choose the transformation to fix (based on the accuracy of the attribute to be modified and the closeness of the new value to the original value). This is followed by applying the transformation to correct the discrepancy.

Missing values

During the process to seize data from all sorts of data sources, there are many cases when some fields are left blank or contain a null value. Good data entry procedures should avoid or minimize the number of missing values or errors. The missing values and defaults are indistinguishable.

If some fields are missing a value, there are a couple of solutions – each with different considerations and shortages and each is applicable within a certain context.

- **Ignore the tuple:** By ignoring the tuple, you cannot make use of the remaining values except the missing one. This method is applicable when the tuple contains several attributes with missing values or the percentage of missing value per attribute doesn't vary considerably.
- **Filling the missing value manually:** This is not applicable for large datasets.
- **Use a global constant to fill the value:** Applying the value to fill the missing value will misguide the mining process, and is not foolproof.

- **Use a measure for a central tendency for the attribute to fill the missing value:** The measures of central tendency can be used for symmetric data distribution.
- **Use the attribute mean or median:** Use the attribute mean or median for all samples belonging to the same class as the given tuple.
- **Use the most probable value to fill the missing value:** The missing data can be filled with data determined with regression, inference-based tool, such as Bayesian formalism or decision tree induction.

The most popular method is the last one; it is based on the present values and values from other attributes.

Junk, noisy data, or outlier

As in a physics or statistics test, noise is a random error that occurs during the test process to seize the measured data. No matter what means you apply to the data gathering process, noise inevitably exists.

Approaches for data smoothing are listed here. Along with the progress of data mining study, new methods keep occurring. Let's have a look at them:

- **Binning:** This is a local scope smoothing method in which the neighborhood values are used to compute the final value for the certain bin. The sorted data is distributed into a number of bins and each value in that bin will be replaced by a value depending on some certain computation of the neighboring values. The computation can be bin median, bin boundary, which is the boundary data of that bin.
- **Regression:** The target of regression is to find the best curve or something similar to one in a multidimensional space; as a result, the other values will be used to predict the value of the target attribute or variable. In other aspects, it is a popular means for smoothing.
- **Classification or outlier:** The classifier is another inherent way to find the noise or outlier. During the process of classifying, most of the source data is grouped into couples of groups, except the outliers.

Data integration

Data integration combines data from multiple sources to form a coherent data store. The common issues here are as follows:

- **Heterogeneous data:** This has no common key
- **Different definition:** This is intrinsic, that is, same data with different definition, such as a different database schema
- **Time synchronization:** This checks if the data is gathered under same time periods
- **Legacy data:** This refers to data left from the old system
- **Sociological factors:** This is the limit of data gathering

There are several approaches that deal with the above issues:

- **Entity identification problem:** Schema integration and object matching are tricky. This referred to as the entity identification problem.
- **Redundancy and correlation analysis:** Some redundancies can be detected by correlation analysis. Given two attributes, such an analysis can measure how strongly one attribute implies the other, based on the available data.
- **Tuple Duplication:** Duplication should be detected at the tuple level to detect redundancies between attributes
- **Data value conflict detection and resolution:** Attributes may differ on the abstraction level, where an attribute in one system is recorded at a different abstraction level

Data dimension reduction

Reduction of dimensionality is often necessary in the analysis of complex multivariate datasets, which is always in high-dimensional format. So, for example, problems modeled by the number of variables present, the data mining tasks on the multidimensional analysis of qualitative data. There are also many methods for data dimension reduction for qualitative data.

The goal of dimensionality reduction is to replace large matrix by two or more other matrices whose sizes are much smaller than the original, but from which the original can be approximately reconstructed, usually by taking their product with loss of minor information.

Eigenvalues and Eigenvectors

An eigenvector for a matrix is defined as when the matrix (A in the following equation) is multiplied by the eigenvector (v in the following equation). The result is a constant multiple of the eigenvector. That constant is the eigenvalue associated with this eigenvector. A matrix may have several eigenvectors.

$$Av = \lambda v$$

An eigenpair is the eigenvector and its eigenvalue, that is, (v, λ) in the preceding equation.

Principal-Component Analysis

The **Principal-Component Analysis (PCA)** technique for dimensionality reduction views data that consists of a collection of points in a multidimensional space as a matrix, in which rows correspond to the points and columns to the dimensions.

The product of this matrix and its transpose has eigenpairs, and the principal eigenvector can be viewed as the direction in the space along which the points best line up. The second eigenvector represents the direction in which deviations from the principal eigenvector are the greatest.

Dimensionality reduction by PCA is to approximate the data by minimizing the root-mean-square error for the given number of columns in the representing matrix, by representing the matrix of points by a small number of its eigenvectors.

Singular-value decomposition

The **singular-value decomposition (SVD)** of a matrix consists of following three matrices:

- U
- Σ
- V

U and V are column-orthonormal; as vectors, the columns are orthogonal and their length is 1. Σ is a diagonal matrix and the values along its diagonal are called singular values. The original matrix equals to the product of U , Σ , and the transpose of V .


SVD is useful when there are a small number of concepts that connect the rows and columns of the original matrix.

Dimensionality reduction by SVD for matrix U and V are typically as large as the original. To use fewer columns for U and V , delete the columns corresponding to the smallest singular values from U , V , and Σ . This minimizes the error in reconstruction of the original matrix from the modified U , Σ , and V .

CUR decomposition

The CUR decomposition seeks to decompose a sparse matrix into sparse, smaller matrices whose product approximates the original matrix.

The CUR chooses from a given sparse matrix a set of columns C and a set of rows R , which play the role of U and V^T in SVD. The choice of rows and columns is made randomly with a distribution that depends on the square root of the sum of the squares of the elements. Between C and R is a square matrix called U , which is constructed by a pseudo-inverse of the intersection of the chosen rows and columns.

 By CUR solution, the three component matrices C , U , and R will be retrieved. The product of those three will approximate the original matrix M . For R community, rCUR is an R package for the CUR matrix decomposition.

Data transformation and discretization

As we know from the previous section, there are always some data formats that are best suited for specific data mining algorithms. Data transformation is an approach to transform the original data to preferable data format for the input of certain data mining algorithms before the processing.

Data transformation

Data transformation routines convert the data into appropriate forms for mining. They're shown as follows:

- **Smoothing:** This uses binning, regression, and clustering to remove noise from the data
- **Attribute construction:** In this routine, new attributes are constructed and added from the given set of attributes
- **Aggregation:** In this summary or aggregation, operations are performed on the data

- **Normalization:** Here, the attribute data is scaled so as to fall within a smaller range
- **Discretization:** In this routine, the raw values of a numeric attribute are replaced by interval label or conceptual label
- **Concept hierarchy generation for nominal data:** Here, attributes can be generalized to higher level concepts

Normalization data transformation methods

To avoid dependency on the choice of measurement units on data attributes, the data should be normalized. This means transforming or mapping the data to a smaller or common range. All attributes gain an equal weight after this process. There are many normalization methods. Let's have a look at some of them:

- **Min-max normalization:** This preserves the relationships among the original data values and performs a linear transformation on the original data. The applicable ones of the actual maximum and minimum values of an attribute will be normalized.
- **z-score normalization:** Here the values for an attribute are normalized based on the mean and standard deviation of that attribute. It is useful when the actual minimum and maximum of an attribute to be normalized are unknown.
- **Normalization by decimal scaling:** This normalizes by moving the decimal point of values of attribute.

Data discretization

Data discretization transforms numeric data by mapping values to interval or concept labels. Discretization techniques include the following:

- **Data discretization by binning:** This is a top-down unsupervised splitting technique based on a specified number of bins.
- **Data discretization by histogram analysis:** In this technique, a histogram partitions the values of an attribute into disjoint ranges called buckets or bins. It is also an unsupervised method.
- **Data discretization by cluster analysis:** In this technique, a clustering algorithm can be applied to discretize a numerical attribute by partitioning the values of that attribute into clusters or groups.

- **Data discretization by decision tree analysis:** Here, a decision tree employs a top-down splitting approach; it is a supervised method. To discretize a numeric attribute, the method selects the value of the attribute that has minimum entropy as a split-point, and recursively partitions the resulting intervals to arrive at a hierarchical discretization.
- **Data discretization by correlation analysis:** This employs a bottom-up approach by finding the best neighboring intervals and then merging them to form larger intervals, recursively. It is supervised method.

Visualization of results

Visualization is the graphic presentation of data-portrayals meant to reveal complex information at a glance, referring to all types of structured representation of information. This includes graphs, charts, diagrams, maps, storyboards, and other structured illustrations.

Good visualization of results gives you the chance to look at data through the eyes of experts. It is beautiful not only for their aesthetic design, but also for the elegant layers of detail that efficiently generate insight and new understanding.

The result of every data mining algorithm can be visualized and clarified by the use of the algorithms. Visualization plays an important role in the data mining process.

There are four major features that create the best visualizations:

- **Novel:** It must not only merely being a conduit for information, but offer some novelty in the form of new style of information.
- **Informative:** The attention to these factors and the data itself will make a data visualization effective, successful, and beautiful.
- **Efficient:** A nice visualization has an explicit goal, a clearly defined message, or a special perspective on the information that it is made to convey. It must be as simple as possible and straightforward, but shouldn't lose out on necessary, relevant complexity. The irrelevant data serves as noises here. It should reflect the qualities of the data that they represent, reveal properties and relationships inherent and implicit in the data source to bring new knowledge, insight, and enjoyment to final user.
- **Aesthetic:** The graphic must serve the primary goal of presenting information, not only axes and layout, shapes, lines, and typography, but also the appropriate usage of these ingredients.

Visualization with R

R provides the production of publication-quality diagrams and plots. There are graphic facilities distributed with R, and also some facilities that are not part of the standard R installation. You can use R graphics from command line.

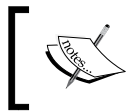
The most important feature of the R graphics setup is the existence of two distinct graphics systems within R:

- The traditional graphics system
- Grid graphics system

The most appropriate facilities will be evaluated and applied to the visualization of every result of all algorithms listed in the book.

Functions in the graphics systems and add-on packages can be divided into several types:

- High-level functions that produce complete plots
- Low-level functions to add further output to an existing plot
- The ones to work interactively with graphical output



R graphics output can be produced in a wide range of graphical formats, such as PNG, JPEG, BMP, TIFF, SVG, PDF, and PS.

To enhance your knowledge about this chapter, here are some practice questions for you to have check about the concepts.

Time for action

Let's now test what we've learned so far:

- What is the difference between data mining and machine learning?
- What is data preprocessing and data quality?
- Download R and install R on your machine.
- Compare and contrast data mining and machine learning.

Summary

In this chapter, we looked at the following topics:

- An introduction to data mining and available data sources
- A quick overview of R and the necessity to use R
- A description of statistics and machine learning, and their relations to data mining
- The two standard industrial data mining process
- Data attributes types and the data measurement approaches
- The three important steps in data preprocessing
- An introduction to the scalability and efficiency of data mining algorithms, and data visualization methods and necessities
- A discussion on social network mining, text mining, and web data mining
- A short introduction about RHadoop and Map Reduce

In the following chapters, the reader will learn how to implement various data mining algorithms and manipulate data with R.

2

Mining Frequent Patterns, Associations, and Correlations

In this chapter, we will learn how to mine frequent patterns, association rules, and correlation rules when working with R programs. Then, we will evaluate all these methods with benchmark data to determine the interestingness of the frequent patterns and rules. We will cover the following topics in this chapter:

- Introduction to associations and patterns
- Market basket analysis
- Hybrid association rules mining
- Mining sequence datasets
- High-performance algorithms

The algorithms to find frequent items from various data types can be applied to numeric or categorical data. Most of these algorithms have one common basic algorithmic form, which is **A-Priori**, depending on certain circumstances. Another basic algorithm is **FP-Growth**, which is similar to A-Priori. Most pattern-related mining algorithms derive from these basic algorithms.

With frequent patterns found as one input, many algorithms are designed to find association and correlation rules. Each algorithm is only a variation from the basic algorithm.

Along with the growth, size, and types of datasets from various domains, new algorithms are designed, such as the multistage algorithm, the multihash algorithm, and the limited-pass algorithm.

An overview of associations and patterns

One popular task for data mining is to find relations among the source dataset; this is based on searching frequent patterns from various data sources, such as market baskets, graphs, and streams.

All the algorithms illustrated in this chapter are written from scratch in the R language for the purpose of explaining association analysis, and the code will be demonstrated using the standard R packages for the algorithms such as arules.

Patterns and pattern discovery

With many applications across a broad field, frequent pattern mining is often used in solving various problems, such as the market investigation for a shopping mall from the transaction data.

Frequent patterns are the ones that often occur in the source dataset. The dataset types for frequent pattern mining can be itemset, subsequence, or substructure. As a result, the frequent patterns found are known as:

- Frequent itemset
- Frequent subsequence
- Frequent substructures

These three frequent patterns will be discussed in detail in the upcoming sections.

These newly founded frequent patterns will serve as an important platform when searching for recurring interesting rules or relationships among the given dataset.

Various patterns are proposed to improve the efficiency of mining on a dataset. Some of them are as follows; they will be defined in detail later:

- Closed patterns
- Maximal patterns
- Approximate patterns
- Condensed patterns
- Discriminative frequent patterns

The frequent itemset

The **frequent itemset** originated from true market basket analysis. In a store such as Amazon, there are many orders or transactions; a certain customer performs a transaction where their Amazon shopping cart includes some items. The mass result of all customers' transactions can be used by the storeowner to find out what items are purchased together by customers. As a simple definition, itemset denotes a collection of zero or more items.

We call a transaction a **basket**, and a set of items can belong to any basket. We will set the variable s as the support threshold, which is compared with the count of a certain set of items that appear in all the baskets. If the count of a certain set of items that appear in all the baskets is not less than s , we would call the itemset a **frequent itemset**.

An itemset is called a k -itemset if it contains k pieces of items, where k is a non-zero integer. The support count of an itemset is $support_count(X)$, the count of itemset contained X , given the dataset.

For a predefined minimum support threshold s , the itemset x is a frequent itemset if $support_count(X) \geq s$. The minimum support threshold s is a customizable parameter, which can be adjusted by domain experts or experiences.

The frequent itemset is also used in many domains. Some of them are shown in the following table:

| | Items | Baskets | Comments |
|-------------------------|-------------------------|---------------------------------|----------|
| Related concepts | Words | Documents | |
| Plagiarism | Documents | Sentences | |
| Biomarkers | Biomarkers and diseases | The set of data about a patient | |

If an itemset is frequent, then any of its subset must be frequent. This is known as the A-Priori principle, the foundation of the A-Priori algorithm. The direct application of the A-Priori principle is to prune the huge number of frequent itemsets.

One important factor that affects the number of frequent itemsets is the minimum support count: the lower the minimum support count, the larger the number of frequent itemsets.

For the purpose of optimizing the frequent itemset-generation algorithm, some more concepts are proposed:

- An itemset X is closed in dataset S , if $\forall Y \in S, X \subset Y, \text{then } support_count(X) \neq support_count(Y)$; X is also called a closed itemset. In other words, if X is frequent, then X is a closed frequent itemset.
- An itemset X is a maximal frequent itemset if $\forall Y \in S, X \subset Y, \text{then } Y \text{ is not frequent}$; in other words, Y does not have frequent supersets.
- An itemset X is considered a constrained frequent itemset once the frequent itemset satisfies the user-specified constraints.
- An itemset X is an approximate frequent itemset if X derives only approximate support counts for the mined frequent itemsets.
- An itemset X is a top- k frequent itemset in the dataset S if X is the k -most frequent itemset, given a user-defined value k .

The following example is of a transaction dataset. All itemsets only contain items from the set, $D = \{I_k | \{I_k | k \in [1, 7]\}$. Let's assume that the minimum support count is 3.

| tid (transaction id) | List of items in the itemset or transaction |
|----------------------|---|
| T001 | I_1, I_2, I_4, I_7 |
| T002 | I_2, I_3, I_6 |
| T003 | I_1, I_4, I_6 |
| T004 | I_1, I_2, I_5 |
| T005 | I_2, I_3, I_4 |
| T006 | I_2, I_5, I_6 |
| T007 | I_2, I_4, I_7 |
| T008 | I_1, I_7 |
| T009 | I_1, I_2, I_3 |
| T010 | I_1, I_2, I_4 |

Then, we will get the frequent itemsets $L_1 = \{I_k | k \in \{1, 2, 4, 6, 7\}\}$ and $L_2 = \{\{I_1, I_2\}, \{I_1, I_4\}, \{I_2, I_4\}\}$.

The frequent subsequence

The frequent sequence is an ordered list of elements where each element contains at least one event. An example of this is the page-visit sequence on a site by the specific web page the user is on more concretely speaking, the order in which a certain user visits web pages. Here are two examples of the frequent subsequence:

- **Customer:** Successive shopping records of certain customers in a shopping mart serves as the sequence, each item bought serves as the event item, and all the items bought by a customer in one shopping are treated as elements or transactions
- **Web usage data:** Users who visit the history of the WWW are treated as a sequence, each UI/page serves as the event or item, and the element or transaction can be defined as the pages visited by users with one click of the mouse

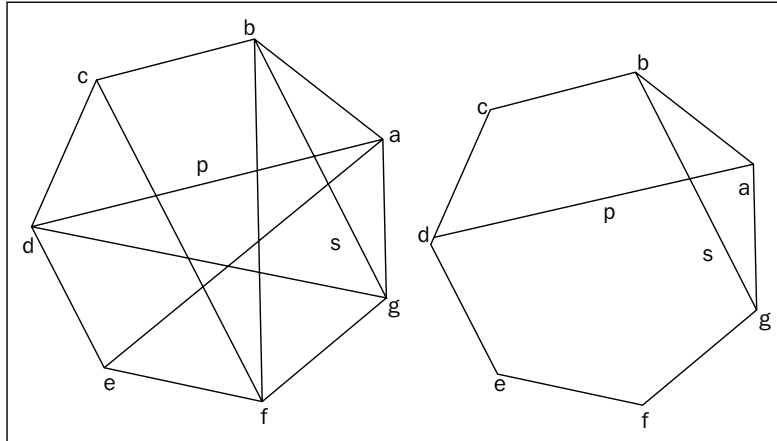
The length of a sequence is defined by the number of items contained in the sequence. A sequence of length k is called a k -sequence. The size of a sequence is defined by the number of itemsets in the sequence. We call a sequence $s_1 = \langle a_1 a_2 \dots a_r \rangle$ as a subsequence of the sequence $s_2 = \langle b_1 b_2 \dots b_r \rangle$ or s_2 as the super sequence of s_1 when $\exists 1 \leq j_1 \leq j_2 \leq \dots \leq j_{r-1} \leq j_r \leq v$, and $a_1 \sqsubseteq b_{j_1}, a_2 \sqsubseteq b_{j_2}, \dots, a_r \sqsubseteq b_{j_r}$ is satisfied.

The frequent substructures

In some domains, the tasks under research can be modeled with a graph theory. As a result, there are requirements for mining common subgraphs (subtrees or sublattices); some examples are as follows:

- **Web mining:** Web pages are treated as the vertices of graph, links between pages serve as edges, and a user's page-visiting records construct the graph.
- **Network computing:** Any device with computation ability on the network serves as the vertex, and the interconnection between these devices serves as the edge. The whole network that is made up of these devices and interconnections is treated as a graph.
- **Semantic web:** XML elements serve as the vertices, and the parent/child relations between them are edges; all these XML files are treated as graphs.

A graph G is represented by $G = (V, E)$, where V represents a group of vertices, and E represents a group of edges. A graph $G' = (V', E')$ is called as subgraph of graph $G = (V, E)$ once $V' \subseteq V$ and $E' \subseteq E$. Here is an example of a subgraph. There is the original graph with vertices and edges on the left-hand side of the following figure and the subgraph on the right-hand side with some edges omitted (or omission of vertices in other circumstances):



Relationship or rules discovery

Mining of association rules is based on the frequent patterns found. The different emphases on the interestingness of relations derives two types of relations for further research: association rules and correlation rules.

Association rules

In a later section, a method to show association analysis is illustrated; this is a useful method to discover interesting relationships within a huge dataset. The relations can be represented in the form of association rules or frequent itemsets.

Association rule mining is to find the result rule set on a given dataset (the transaction data set or other sequence-pattern-type dataset), a predefined minimum support count s , and a predefined confidence c , given any found rule $X \rightarrow Y$ $\text{support_count}(X \rightarrow Y) \geq s$, and $\text{confidence}(X \rightarrow Y) \geq c$.

$X \rightarrow Y$ is an association rule where $X \cap Y = \phi$; X and Y are disjoint. The interesting thing about this rule is that it is measured by its **support** and **confidence**. Support means the frequency in which this rule appears in the dataset, and confidence means the probability of the appearance of Y when X is present.

For association rules, the key measures of rule interestingness are rule support and confidence. Their relationship is given as follows:

$$\text{confidence}(X \rightarrow Y) = P(Y|X) = \frac{P(X \cup Y)}{P(X)} = \frac{\text{support_count}(X \cup Y)}{\text{support_count}(X)}$$

$\text{support_count}(x)$ is the count of itemset in the dataset, contained x .

As a convention, in $\text{support_count}(x)$, in the confidence value and support count value are represented as a percentage between 0 and 100.

The association rule $X \rightarrow Y$ is strong once $\text{confidence}(X \rightarrow Y) \geq c$ and $\text{support_count}(X \cup Y) > s$. The predefined minimum support threshold is s , and c is the predefined minimum confidence threshold.

The meaning of the found association rules should be explained with caution, especially when there is not enough to judge whether the rule implies causality. It only shows the co-occurrence of the prefix and postfix of the rule. The following are the different kinds of rules you can come across:

- A rule is a Boolean association rule if it contains association of the presence of the item
- A rule is a single-dimensional association if there is, at the most, only one dimension referred to in the rules
- A rule is a multidimensional association rule if there are at least two dimensions referred to in the rules
- A rule is a correlation-association rule if the relations or rules are measured by statistical correlation, which, once passed, leads to a correlation rule
- A rule is a quantitative-association rule if at least one item or attribute contained in it is quantitative

Correlation rules

In some situations, the support and confidence pairs are not sufficient to filter uninteresting association rules. In such a case, we will use support count, confidence, and correlations to filter association rules.

There are a lot of methods to calculate the correlation of an association rule, such as χ^2 analyses, all-confidence analysis, and cosine. For a k-itemset $X = \{i_1, i_2, \dots, i_k\}$, define the all-confidence value of X as:

$$all_confidence(X) = support_count(X) / \max\{support_count(i_j) \mid \forall i_j \in X\}$$

$$lift(X \rightarrow Y) = \frac{confidence(X \rightarrow Y)}{P(Y)} = P(X \cup Y) / (P(X)P(Y))$$

Market basket analysis

Market basket analysis is the methodology used to mine a shopping cart of items bought or just those kept in the cart by customers. The concept is applicable to a variety of applications, especially for store operations. The source dataset is a massive data record. The aim of market basket analysis is to find the association rules between the items within the source dataset.

The market basket model

The market basket model is a model that illustrates the relation between a basket and its associated items. Many tasks from different areas of research have this relation in common. To summarize them all, the market basket model is suggested as the most typical example to be researched.

The basket is also known as the transaction set; this contains the itemsets that are sets of items belonging to same itemset.

The A-Priori algorithm is a level wise, itemset mining algorithm. The Eclat algorithm is a tidset intersection itemset mining algorithm based on tidset intersection in contrast to A-Priori. FP-growth is a frequent pattern tree algorithm. The tidset denotes a collection of zeros or IDs of transaction records.

A-Priori algorithms

As a common strategy to design algorithms, the problem is divided into two subproblems:

- The frequent itemset generation
- Rule generation

The strategy dramatically decreases the search space for association mining algorithms.

Input data characteristics and data structure

As the input of the A-Priori algorithm, the original input itemset is binarized, that is, 1 represents the presence of a certain item in the itemset; otherwise, it is 0. As a default assumption, the average size of the itemset is small. The popular preprocessing method is to map each unique available item in the input dataset to a unique integer ID.

The itemsets are usually stored within databases or files and will go through several passes. To control the efficiency of the algorithm, we need to control the count of passes. During the process when itemsets pass through other itemsets, the representation format for each itemset you are interested in is required to count and store for further usage of the algorithm.


There is a monotonicity feature in the itemsets under research; this implies that every subset of a frequent itemset is frequent. This characteristic is used to prune the search space for the frequent itemset in the process of the A-Priori algorithm. It also helps compact the information related to the frequent itemset. This feature gives us an intrinsic view that focuses on smaller-sized frequent itemsets. For example, there are three frequent 2-itemsets contained by one certain frequent 3-itemset.

 When we talk about k-itemsets means an itemset containing k items.

The basket is in a format called the **horizontal format** and contains a basket or transaction ID and a number of items; it is used as the basic input format for the A-Priori algorithm. In contrast, there is another format known as the **vertical format**; this uses an item ID and a series of the transaction IDs. The algorithm that works on vertical data format is left as an exercise for you.

The A-Priori algorithm

Two actions are performed in the generation process of the A-Priori frequent itemset: one is **join**, and the other is **prune**.

 One important assumption is that the items within any itemset are in a lexicographic order.

- **Join action:** Given that L_k is the set of frequent k-itemsets, a set of candidates to find L_{k+1} is generated. Let's call it C_{k+1} .

$$\begin{aligned}
 C_k &= L_{k-1} \bowtie L_{k-1} = \{l' \mid \exists l_1, l_2 \in L_{k-1}, \forall m \in [1, k-2], l_1[m] \\
 &= l_2[m] \text{ and } l_1[k-1] \leq l_2[k-1], \\
 &\text{then } l' < l_1[1], l_1[2], \dots, l_1[k-2], l_1[k-1], l_2[k-1] \}
 \end{aligned}$$

- **Prune action:** $L_k \subseteq C_k$, the size of C_k , the candidate itemset, is usually much bigger than L_k , to save computation cost; monotonicity characteristic of frequent itemset is used here to prune the size of C_k .

$$\forall c \in C_k, \exists c_{k-1} \text{ is a } (k-1)\text{-subset of } c, \text{ and } c_{k-1} \notin L_{k-1} \Rightarrow c \notin L_k$$

Here is the pseudocode to find all the frequent itemsets:

```

APRIORI (D, I, minsup)
   $\mathcal{F} \leftarrow \emptyset$ 
   $\mathcal{C}^{(1)} \leftarrow \{\emptyset\}$ 
  foreach  $i \in \mathcal{I}$  do Add  $i$  as child of  $\emptyset$  in  $\mathcal{C}^{(1)}$  with  $sup(i) \leftarrow 0$ 
   $k \leftarrow 1$ 
  while  $\mathcal{C}^{(k)} \neq \emptyset$  do
    COMPUTESUPPORT ( $\mathcal{C}^{(k)}$ , D)
    foreach leaf  $X \in \mathcal{C}^{(k)}$  do
      if  $sup(X) \geq minsup$  then  $\mathcal{F} \leftarrow \mathcal{F} \cup \{(X, sup(X))\}$ 
      else remove  $X$  from  $\mathcal{C}^{(k)}$ 
     $\mathcal{C}^{(k+1)} \leftarrow$  EXTENDPREFIXTREE ( $\mathcal{C}^{(k)}$ )
     $k \leftarrow k + 1$ 
  return  $\mathcal{F}^{(k)}$ 

```

```

COMPUTESUPPORT ( $\mathcal{C}^{(k)}$ , D):
  foreach  $\langle t, i(t) \rangle \in \mathbf{D}$  do
    foreach  $k$ -subset  $X \subseteq i(t)$  do
      if  $X \in \mathcal{C}^{(k)}$  then  $sup(X) \leftarrow sup(X) + 1$ 

```

```

EXTENDPREFIXTREE ( $\mathcal{C}^{(k)}$ ):
foreach leaf  $X_a \in \mathcal{C}^{(k)}$  do
  foreach leaf  $X_b \in \text{SIBLING}(X_a)$ , such that  $b > a$  do
     $X_{ab} \leftarrow X_a \cup X_b$ 

    if  $X_j \in \mathcal{C}^{(k)}$ , for all  $X_j \subset X_{ab}$ , such that  $|X_j| = |X_{ab}| - 1$  then
      Add  $X_{ab}$  as child of  $X_a$  with  $\text{sup}(X_{ab}) \leftarrow 0$ 

    if no extensions from  $X_a$  then remove  $X_a$  from  $\mathcal{C}^{(k)}$ 
return  $\mathcal{C}^{(k)}$ 

```

The R implementation

R code of the A-Priori frequent itemset generation algorithm goes here. D is a transaction dataset. Suppose `MIN_SUP` is the minimum support count threshold. The output of the algorithm is L, which is a frequent itemsets in D.


The output of the A-Priori function can be verified with the R add-on package, `arules`, which is a pattern-mining and association-rules-mining package that includes A-Priori and `éclat` algorithms. Here is the R code:

```

Apriori <- function (data, I, MIN_SUP, parameter = NULL){
  f <- CreateItemsets()
  c <- FindFrequentItemset (data,I,1, MIN_SUP)
  k <- 2
  len4data <- GetDatasetSize (data)
  while( !IsEmpty(c[[k-1]]) ){
    f[[k]] <- AprioriGen(c[k-1])
    for( idx in 1: len4data ){
      ft <- GetSubSet (f[[k]],data[[idx]])
      len4ft <- GetDatasetSize(ft)
      for( jdx in 1:len4ft ){
        IncreaseSupportCount (f[[k]],ft[jdx])
      }
    }
    c[[k]] <- FindFrequentItemset (f[[k]],I,k,MIN_SUP)
    k <- k+1
  }
  c
}

```

To verify the R code, the `arules` package is applied while verifying the output.

 Arules (Hahsler et al., 2011) provides the support to mine frequent itemsets, maximal frequent itemsets, closed frequent itemsets, and association rules too. A-Priori and Eclat algorithms are both available. Also cSPADE can be found in arulesSequence, the add-on for arules.

Given:

$$D = \{tinnedfruit, tuna, milk, coke, water, biscuits, oil, soap\}$$

At first, we will sort D into an ordered list in a predefined order algorithm or simply the natural order of characters, which is used here. Then:

$$D = \left\{ \begin{array}{l} I1 = biscuits, I2 = coke, I3 = milk, I4 = oil, \\ I5 = soap, I6 = tinnedfruit, I7 = tuna \end{array} \right\} = \{I_k | k \in [1, 7]\}$$

Let's assume that the minimum support count is 5; the following table is an input dataset:

| tid (transaction id) | List of items in the itemset or transaction |
|----------------------|---|
| T001 | I_1, I_2, I_4, I_7 |
| T002 | I_2, I_3, I_6 |
| T003 | I_1, I_4, I_6 |
| T004 | I_1, I_2, I_5 |
| T005 | I_2, I_3, I_4 |
| T006 | I_2, I_5, I_6 |
| T007 | I_2, I_4, I_7 |
| T008 | I_1, I_7 |
| T009 | I_1, I_2, I_3 |
| T010 | I_1, I_2, I_4 |

In the first scan or pass of the dataset D , get the count of each candidate itemset C_1 .
The candidate itemset and its related count:

| Itemset | Support count |
|-----------|---------------|
| $\{I_1\}$ | 6 |
| $\{I_2\}$ | 8 |
| $\{I_3\}$ | 2 |
| $\{I_4\}$ | 5 |
| $\{I_5\}$ | 2 |
| $\{I_6\}$ | 3 |
| $\{I_7\}$ | 3 |

We will get the L_1 after comparing the support count with minimum support count.

| Itemset | Support count |
|-----------|---------------|
| $\{I_1\}$ | 6 |
| $\{I_2\}$ | 8 |
| $\{I_4\}$ | 5 |

We will generate C_2 by L_1 , $C_2 = \{\{I_1, I_2\}, \{I_1, I_4\}, \{I_2, I_4\}\}$.

| Itemset | Support count |
|----------------|---------------|
| $\{I_1, I_2\}$ | 4 |
| $\{I_1, I_4\}$ | 3 |
| $\{I_2, I_4\}$ | 4 |

After comparing the support count with the minimum support count, we will get $L_2 = \phi$. The algorithm then terminates.

A-Priori algorithm variants

The various variants of A-Priori algorithms are designed mainly for the purpose of efficiency and scalability. Some of the improvements of the A-Priori algorithms are discussed in the upcoming sections.

The Eclat algorithm

The A-Priori algorithm loops as many times as the maximum length of the pattern somewhere. This is the motivation for the **Equivalence CLASS Transformation (Eclat)** algorithm. The Eclat algorithm explores the vertical data format, for example, using $\langle \text{item id}, \text{tid set} \rangle$ instead of $\langle \text{tid}, \text{item id set} \rangle$ that is, with the input data in the vertical format in the sample market basket file, or to discover frequent itemsets from a transaction dataset. The A-Priori property is also used in this algorithm to get frequent $(k+1)$ itemsets from k itemsets.

The candidate itemset is generated by set intersection. The vertical format structure is called a tidset as defined earlier. If all the transaction IDs related to the item I are stored in a vertical format transaction itemset, then the itemset is the tidset of the specific item.

The support count is computed by the intersection between tidsets. Given two tidsets, X and Y , $\text{support_count}(X \cap Y)$ is the cardinality of $X \cap Y$. The pseudocode is $F \leftarrow \phi$, $P \leftarrow \{ \langle i, t(i) \rangle \mid i \in I, |t(i)| \geq \text{MIN_SUP} \}$.

```

ECLAT ( $P$ ,  $\text{minsup}$ ,  $\mathcal{F}$ ):
  foreach  $\langle X_a, \mathbf{t}(X_a) \rangle \in P$  do
     $\mathcal{F} \leftarrow \mathcal{F} \cup \{ \langle X_a, \text{sup}(X_a) \rangle \}$ 
     $P_a \leftarrow \emptyset$ 
    foreach  $\langle X_b, \mathbf{t}(X_b) \rangle \in P$ , with  $X_b > X_a$  do
       $X_{ab} = X_a \cup X_b$ 
       $\mathbf{t}(X_{ab}) = \mathbf{t}(X_a) \cap \mathbf{t}(X_b)$ 
      if  $\text{sup}(X_{ab}) \geq \text{minsup}$  then
         $P_a \leftarrow P_a \cup \{ \langle X_{ab}, \mathbf{t}(X_{ab}) \rangle \}$ 
  if  $P_a \neq \emptyset$  then ECLAT ( $P_a$ ,  $\text{minsup}$ ,  $\mathcal{F}$ )
  
```

The R implementation

Here is the R code for the Eclat algorithm to find the frequent patterns. Before calling the function, `f` is set to empty, and `p` is the set of frequent 1-itemsets:

```
Eclat <- function (p,f,MIN_SUP){
  len4tidsets <- length(p)
  for(idx in 1:len4tidsets){
    AddFrequentItemset (f,p[[idx]],GetSupport (p[[idx]]))
    Pa <- GetFrequentTidSets (NULL,MIN_SUP)
    for(jdx in idx:len4tidsets){
      if(ItemCompare(p[[jdx]],p[[idx]]) > 0){
        xab <- MergeTidSets (p[[idx]],p[[jdx]])
        if(GetSupport (xab)>=MIN_SUP){
          AddFrequentItemset (pa,xab,
            GetSupport (xab))
        }
      }
    }
  }
  if(!IsEmptyTidSets (pa)){
    Eclat (pa, f, MIN_SUP)
  }
}
```

Here is the running result of one example, $I = \{\text{beer, chips, pizza, wine}\}$. The transaction dataset with horizontal and vertical formats, respectively, are shown in the following table:

| tid | X |
|-----|---------------------|
| 1 | {beer, chips, wine} |
| 2 | {beer, chips} |
| 3 | {pizza, wine} |
| 4 | {chips, pizza} |

| x | tidset |
|-------|---------|
| beer | {1,2} |
| chips | {1,2,4} |
| pizza | {3,4} |
| wine | {1,3} |

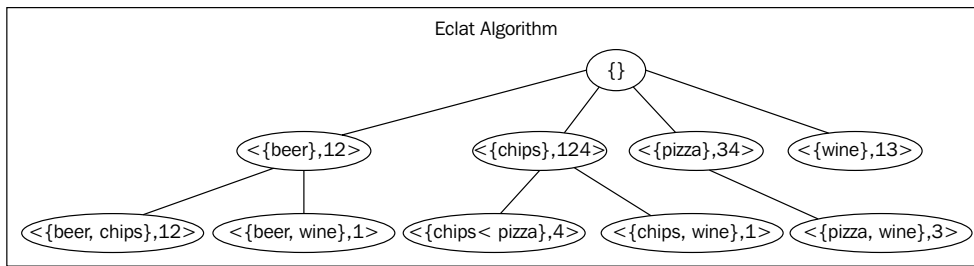
The binary format of this information is in the following table.

| tid | beer | chips | pizza | wine |
|-----|------|-------|-------|------|
| 1 | 1 | 1 | 0 | 1 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 1 | 0 |

Before calling the Eclat algorithm, we will set $MIN_SUP=2$, $F \leftarrow \{\}$,

$$P \leftarrow \{ \langle beer, 12 \rangle, \langle chips, 124 \rangle, \langle pizza, 34 \rangle, \langle wine, 13 \rangle \}$$

The running process is illustrated in the following figure. After two iterations, we will get frequent tidsets, $\{ \langle beer, 12 \rangle, \langle chips, 124 \rangle, \langle pizza, 34 \rangle, \langle wine, 13 \rangle, \langle \{beer, chips\}, 12 \rangle \}$:



The output of the Eclat function can be verified with the R add-on package, arules.

The FP-growth algorithm

The FP-growth algorithm is an efficient method targeted at mining frequent itemsets in a large dataset. The main difference between the FP-growth algorithm and the A-Priori algorithm is that the generation of a candidate itemset is not needed here. The pattern-growth strategy is used instead. The FP-tree is the data structure.

Input data characteristics and data structure

The data structure used is a hybrid of vertical and horizontal datasets; all the transaction itemsets are stored within a tree structure. The tree structure used in this algorithm is called a frequent pattern tree. Here is example of the generation of the structure, $I = \{A, B, C, D, E, F\}$; the transaction dataset D is in the following table, and the FP-tree building process is shown in the next upcoming image. Each node in the FP-tree represents an item and the path from the root to that item, that is, the node list represents an itemset. The support information of this itemset is included in the node as well as the item too.

| tid | X |
|-----|-----------------|
| 1 | {A, B, C, D, E} |
| 2 | {A, B, C, E} |
| 3 | {A, D, E} |
| 4 | {B, E, D} |
| 5 | {B, E, C} |
| 6 | {E, C, D} |
| 7 | {E, D} |

The sorted item order is listed in the following table:

| item | E | D | C | B | A |
|---------------|---|---|---|---|---|
| support_count | 7 | 5 | 4 | 4 | 3 |

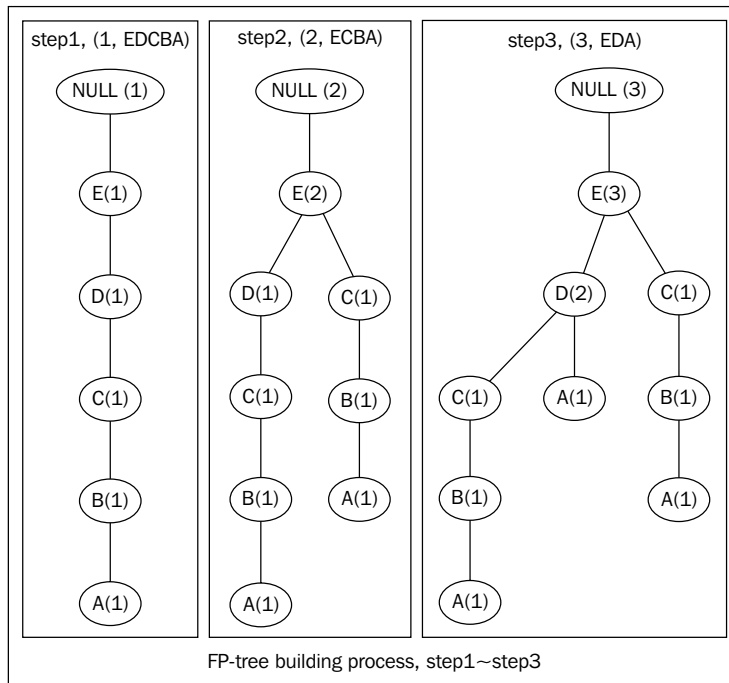
Reorder the transaction dataset with this new decreasing order; get the new sorted transaction dataset, as shown in this table:

| tid | X |
|-----|-----------------|
| 1 | {E, D, C, B, A} |
| 2 | {E, C, B, A} |
| 3 | {E, D, A} |
| 4 | {E, D, B} |
| 5 | {E, C, B} |
| 6 | {E, D, C} |
| 7 | {E, D} |

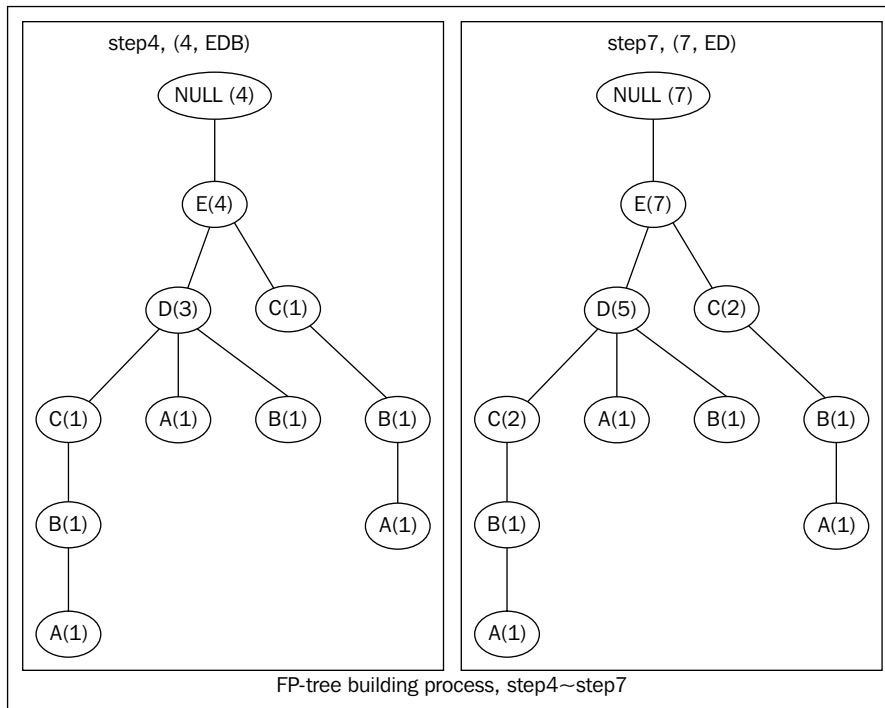
The FP-tree building process is illustrated in the following images, along with the addition of each itemset to the FP-tree. The support information is calculated at the same time, that is, the support counts of the items on the path to that specific node are incremented.

The most frequent items are put at the top of the tree; this keeps the tree as compact as possible. To start building the FP-tree, the items should be decreasingly ordered by the support count. Next, get the list of sorted items and remove the infrequent ones. Then, reorder each itemset in the original transaction dataset by this order.

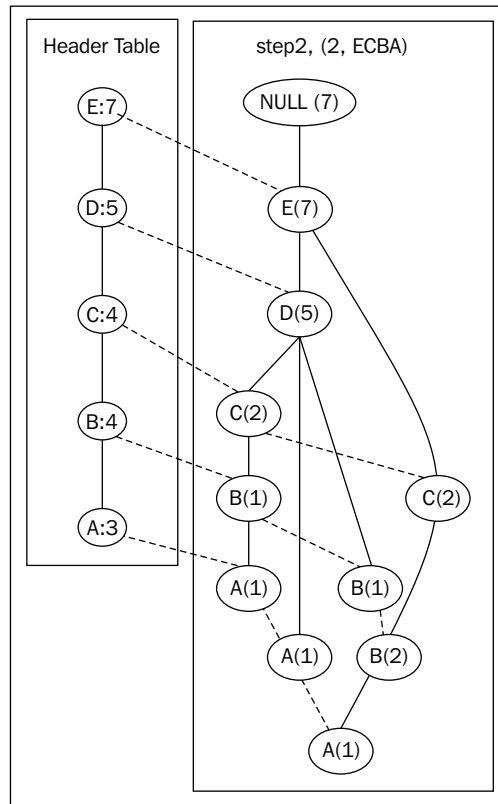
Given $MIN_SUP=3$, the following itemsets can be processed according to this logic:



The result after performing steps 4 and 7 are listed here, and the process of the algorithm is very simple and straight forward:



A header table is usually bound together with the frequent pattern tree. A link to the specific node, or the item, is stored in each record of the header table.



The FP-tree serves as the input of the FP-growth algorithm and is used to find the frequent pattern or itemset. Here is an example of removing the items from the frequent pattern tree in a reverse order or from the leaf; therefore, the order is *A*, *B*, *C*, *D*, and *E*. Using this order, we will then build the projected FP-tree for each item.

The FP-growth algorithm

Here is the pseudocode with recursion definition; the input values are

$R \leftarrow \text{GenerateFPSTree}(D), P \leftarrow \phi, F \leftarrow \phi$

```

 $\mathcal{F}[I] := \{\}$ 
for all  $i \in \mathcal{I}$  occurring in  $\mathcal{D}$  do
   $\mathcal{F}[I] := \mathcal{F}[I] \cup \{I \cup \{i\}\}$ 

 $\mathcal{D}^i := \{\}$ 
 $H := \{\}$ 
for all  $j \in \mathcal{I}$  occurring in  $\mathcal{D}$  such that  $j > i$  do
  if  $\text{support}(I \cup \{i, j\}) \geq \sigma$  then
     $H := H \cup \{j\}$ 
  end if
end for
for all  $(tid, X) \in \mathcal{D}$  with  $i \in X$  do
   $\mathcal{D}^i := \mathcal{D}^i \cup \{(tid, X \cap H)\}$ 
end for
// Depth-first recursion
Compute  $\mathcal{F}[I \cup \{i\}](\mathcal{D}^i, \sigma)$ 
 $\mathcal{F}[I] := \mathcal{F}[I] \cup \mathcal{F}[I \cup \{i\}]$ 
end for

```

The R implementation

Here is the R source code of the main FP-growth algorithm:

```

FPGrowth <- function (r,p,f,MIN_SUP){
  RemoveInfrequentItems (r)
  if (IsPath(r)) {
    y <- GetSubset (r)
    len4y <- GetLength(y)
    for (idx in 1:len4y) {
      x <- MergeSet (p,y[idx])
      SetSupportCount (x, GetMinCnt (x))
      Add2Set (f,x,support_count (x))
    }
  } else {
    len4r <- GetLength(r)
    for (idx in 1:len4r) {
      x <- MergeSet (p,r[idx])
      SetSupportCount (x, GetSupportCount (r[idx]))
      rx <- CreateProjectedFPSTree ()
    }
  }
}

```