DATA MINING FOR A WEB-BASED EDUCATIONAL SYSTEM

By

Behrouz Minaei-Bidgoli

A DISSERTATION

Submitted to

Michigan State University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science and Engineering

2004

ABSTRACT

DATA MINING FOR A WEB-BASED EDUCATIONAL SYSTEM

By

Behrouz Minaei-Bidgoli

Web-based educational technologies allow educators to study how students learn (descriptive studies) and which learning strategies are most effective (causal/predictive studies). Since web-based educational systems are capable of collecting vast amounts of student profile data, data mining and knowledge discovery techniques can be applied to find interesting relationships between attributes of students, assessments, and the solution strategies adopted by students. The focus of this dissertation is three-fold: 1) to introduce an approach for predicting student performance; 2) to use clustering ensembles to build an optimal framework for clustering web-based assessment resources; and 3) to propose a framework for the discovery of interesting association rules within a web-based educational system. Taken together and used within the online educational setting, the value of these tasks lies in improving student performance and the effective design of the online courses.

First, this research presents an approach to classifying student characteristics in order to predict performance on assessments based on features extracted from logged data in a web-based educational system. We show that a significant improvement in classification performance is achieved by using a combination of multiple classifiers. Furthermore, by "learning" an appropriate weighting of the features via a genetic algorithm (GA), we have successfully improved the accuracy of the combined classifier performance by another 10-12%. Such classification is the first step towards a “recommendation system” that will provide valuable, individualized feedback to students.

Second, this project extends previous theoretical work regarding clustering ensembles with the goal of creating an optimal framework for categorizing web-based educational resources. We propose both non-adaptive and adaptive resampling schemes for the integration of multiple clusterings (independent and dependent). Experimental results show improved stability and accuracy for clustering structures obtained via bootstrapping, subsampling, and adaptive techniques. These improvements offer insights into specific associations within the data sets.

Finally, this study turns toward developing a technique for discovering interesting associations between student attributes, problem attributes, and solution strategies. We propose an algorithm for the discovery of “interesting” association rules within a web-based educational system. The main focus is on mining interesting contrast rules, which are sets of conjunctive rules describing interesting characteristics of different segments within a population. In the context of web-based educational systems, contrast rules help to identify attributes characterizing patterns of performance disparity between various groups of students. We propose a general formulation of contrast rules as well as a framework for finding such patterns. Examining these contrasts can improve the online educational systems for both teachers and students – allowing for more accurate assessment and more effective evaluation of the learning process.

This dissertation is dedicated to:

my parents

my wife,

my son Mohsen,

my daughters Maryam and Zahra,

and to whoever serve the Truth for the Truth itself.

Acknowledgements

It is with much appreciation and gratitude that I thank the following individuals who dedicated themselves to the successful completion of this dissertation. Dr. Bill Punch, my major professor and advisor, maintained his post by my side from the inception to the completion of this research project. He gave me guidance throughout my research. Without his knowledge, patience, and support this dissertation would have not been possible. The knowledge and friendship I gained from him will definitely influence the rest of my life.

Other teachers have influenced my thinking and guided my work. I would also thank professor Anil Jain. It was my honor to be a student in his pattern recognition classes, which defined my interest in the field for many years to come. His professional guidance has been a source of inspiration and vital in the completion of this work. I owe many inspirational ideas to Dr. Pang-Ning Tan whose keen insights and valuable discussions often gave me stimulating ideas in research. Though kept busy by his work, he is always willing to share his time and knowledge with me. For these reasons, I wish he would have been at Michigan State University when I began my research.

I also owe many thanks to Dr. Gerd Kortemeyer for his extraordinary support and patience. His productive suggestions and our discussions contributed enormously to this work. From the first time I stepped through the door of Lite Lab in the Division of Science and Mathematics Education, until now, Gerd has allowed me freedom to explore my own directions in research. He supported me materially and morally for many years, and for that I am very grateful. I am grateful to Dr. Esfahanian for taking time to serve on

the guidance committee and overseeing this work. I deeply appreciate his support and insightful suggestions. His great organization of the graduate office in the Computer Science Department helps advance graduate research. I have learned much from him, both as a teacher and as a friend.

I am also grateful to my colleagues in the GARAGe Laboratory especially Alexander Topchy, and all my friends in the LON-CAPA developers group: Helen Keefe, Felicia Berryman, Stuart Raeburn, and Alexander Sakharuk. Discussion with our colleagues Guy Albertelli (LON-CAPA head developer), Matthew Brian Hall, Dr. Edwin Kashy, and Dr. Deborah Kashy were particularly useful. The discussions with them substantially contributed to my work and broadened my knowledge.

I offer my special thanks to Steven Forbes Tuckey in the Writing Center for editing this dissertation and his proof-reading. I am grateful to him for the countless hours of constructive discussion and his informative revisions of my work. Last but not the least, many thanks go to my wife, without her love, care, encouragement and continuous support, I would not be who I am today. Above all, I thank God the Almighty, for blessing me with the strength, health and will to prevail and finish this work.

I and my colleagues in LON-CAPA are thankful to the National Science Foundation for grants supporting this work through the Information Technology Research (ITR 0085921) and the Assessment of Student Achievement (ASA 0243126) programs. Support in earlier years of this project was also received from the Alfred P. Sloan and the Andrew W. Mellon Foundations. We are grateful to our own institution, Michigan State University and to its administrators for over a decade of encouragement and support.

Table of Content

List of Tables

List of Figures

Chapter 1 Introduction

The ever-increasing progress of network-distributed computing and particularly the rapid expansion of the web have had a broad impact on society in a relatively short period of time. Education is on the brink of a new era based on these changes. Online delivery of educational instruction provides the opportunity to bring colleges and universities new energy, students, and revenues. Many leading educational institutions are working to establish an online teaching and learning presence. Several different approaches have been developed to deliver online education in an academic setting. In particular, Michigan State University (MSU) has pioneered some of these systems which provide an infrastructure for online instruction (Multi-Media Physics; CAPA; Lecture*Online;* PhysNet; Kortemeyer and Bauer, 1999; Kashy et al., 1997, LON-CAPA). This study focuses on the data mining aspects of the latest online educational system developed at MSU, the *Learning Online Network with Computer-Assisted Personalized Approach* (*LON-CAPA*).

1.1 Statement of the problem

In LON-CAPA, we are involved with two kinds of large data sets: 1) educational resources such as web pages, demonstrations, simulations, and individualized problems designed for use on homework assignments, quizzes, and examinations; and 2) information about users who create, modify, assess, or use these resources. In other words, we have two ever-growing pools of data. As the resource pool grows, the information from students who have multiple transactions with these resources also increases. The LON-CAPA system logs any access to these resources as well as the sequence and frequency of access in relation to the successful completion of any assignment.

The web browser represents a remarkable enabling tool to get information to and from students. That information can be textual and illustrated, not unlike that presented in a textbook, but also include various simulations representing a modeling of phenomena, essentially experiments on the computer. Its greatest use however is in transmitting information as to the correct or incorrect solutions of various assigned exercises and problems. It also transmits guidance or hints related to the material, sometimes also to the particular submission by a student, and provides the means of communication with fellow students and teaching staff.

This study investigates data mining methods for extracting useful and interesting knowledge from the large database of students who are using LON-CAPA educational resources. This study aims to answer the following research questions:

- How can students be classified based on features extracted from logged data? Do groups of students exist who use these online resources in a similar way? Can we predict for any individual student which group they belong to? Can we use this information to help a student use the resources better, based on the usage of the resource by other students in their groups?

- How can the online problems that students engage in be classified? How do different types of problems impact students' achievements? Can the classifications of problems be employed to find patterns of questions that help student success?
- How can data mining help instructors, problem authors, and course coordinators better design online materials? Can we find sequences of online problems that students use to solve homework problems? Can we help instructors to develop their homework more effectively and efficiently? How can data mining help to detect anomalies in homework problems designed by instructors?
- How can data mining help find patterns of student behavior that groups of students take to solve their problems? Can we find some associative rules between students' educational activities? Can we help instructors predict the approaches that students will take for some types of problems?
- How can data mining be used to identify those students who are at risk, especially in very large classes? Can data mining help the instructor provide appropriate advising in a timely manner?

The goal of this research is to find similar patterns of use in the data gathered from LON-CAPA, and eventually be able to make predictions as to the most beneficial course of studies for each student based on a minimum number of variables for each student. Based on the current state of the student in their learning sequence, the system could then make suggestions as to how to proceed. Furthermore, through clustering of homework problems as well as the sequences that students take to solve those problems, we hope to help instructors design their curricula more effectively. As more and more students enter the online learning environment, databases concerning student access and study patterns will grow. We are going to develop such techniques in order to provide information that can be usefully applied by instructors to increase student learning.

This dissertation is organized as follows: The rest of this first chapter provides basic concepts of data mining and then presents a brief system overview of LON-CAPA that shows how the homework and student data are growing exponentially, while the current statistical measures for analyzing these data are insufficient. Chapter 2 introduces the research background: the important algorithms for data classification and some common clustering methods. Chapter 3 provides information about structure of LON-CAPA data, data retrieval process, representing the statistical information about students, problem and solution strategies, and providing assessment tools in LON-CAPA to detect, to understand, and to address student difficulties. Chapter 4 explains the LON-CAPA experiment to classify students and predict their final grades based on features of their logged data. We design, implement, and evaluate a series of pattern classifiers with various parameters in order to compare their performance in a real dataset from the LON-CAPA system. Results of individual classifiers, and their combination as well as error estimates are presented. Since LON-CAPA data are distributed among several servers and distributed data mining requires efficient algorithms form multiple sources and features, chapter 5 represents a framework for clustering ensembles in order to provide an optimal framework for categorizing distributed web-based educational resources. Chapter 6 discusses the methods to find interesting association rules within the students' databases. We propose a framework for the discovery of interesting association rules within a web-based educational system. Taken together and used within the online educational setting, the value of these tasks lies in improving student performance and the effective design of the online courses. Chapter 7 presents the conclusion of the proposal and discusses the importance of future work.

1.2 Data Mining

Presently, the amount of data stored in databases is increasing at a tremendous speed. This gives rise to a need for new techniques and tools to aid humans in automatically and intelligently analyzing huge data sets to gather useful information. This growing need gives birth to a new research field called Knowledge Discovery in Databases (KDD) or Data Mining, which has attracted attention from researchers in many different fields including database design, statistics, pattern recognition, machine learning, and data visualization. In this chapter we give a definition of KDD and Data Mining, describing its tasks, methods, and applications. Our motivation in this study is gaining the best technique for extracting useful information from large amounts of data in an online educational system, in general, and from the LON-CAPA system, in particular. The goals for this study are: to obtain an optimal predictive model for students within such systems, help students use the learning resources better, based on the usage of the resource by other students in their groups, help instructors design their curricula more effectively, and provide the information that can be usefully applied by instructors to increase student learning.

1.2.1 What is Data Mining?

Data Mining is the process of analyzing data from different perspectives and summarizing the results as useful information. It has been defined as "*the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data*" (Frawley et al., 1992; Fayyad et al., 1996).

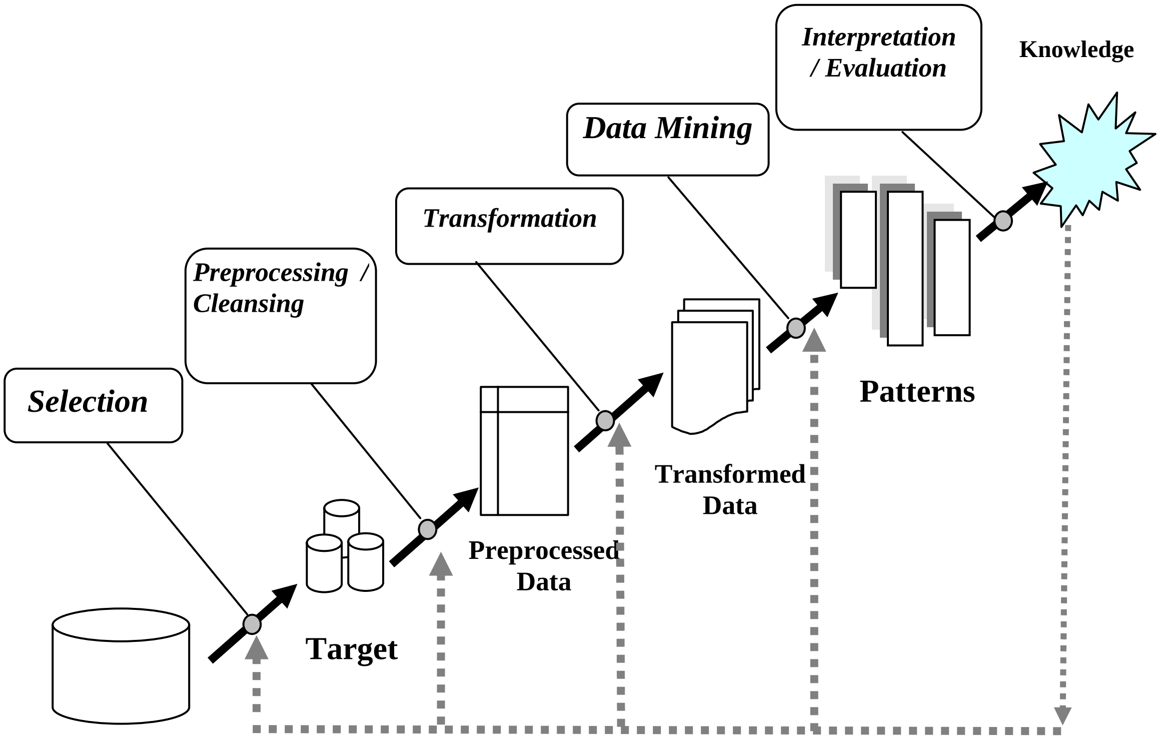


Figure 1.1 Steps of the KDD Process (Fayyad et al., 1996)

The process of data mining uses machine learning, statistics, and visualization techniques to discover and present knowledge in a form that is easily comprehensible. The word "*Knowledge"* in KDD refers to the discovery of patterns which are extracted from the processed data. A *pattern* is an expression describing facts in a subset of the data. Thus, the difference between KDD and data mining is that

> "*KDD refers to the overall process of discoverying knowledge from data while data mining refers to application of algorithms for extracting patterns from data without the additional steps of the KDD process.*" (Fayyad et al., 1996)

However, since Data Mining is a crucial and important part of the KDD process, most researchers use both terms interchangeably. Figure 1.1 presents the iterative nature of the KDD process. Here we outline some of its basic steps as are mentioned in Brachman & Anad (1996):

- Providing an understanding of the application domain, the goals of the system and its users, and the relevant prior background and prior knowledge (This step in not specified in this figure.)
- Selecting a data set, or focusing on a subset of variables or data samples, on which discovery is to be performed
- Preprocessing and data cleansing, removing the noise, collecting the necessary information for modeling, selecting methods for handling missing data fields, accounting for time sequence information and changes
- Data reduction and projection, finding appropriate features to represent data, using dimensionality reduction or transformation methods to reduce the number of variables to find invariant representations for data
- Choosing the data mining task depending on the goal of KDD: clustering, classification, regression, and so forth

- Selecting methods and algorithms to be used for searching for the patterns in the data
- Mining the knowledge: searching for patterns of interest
- Evaluating or interpreting the mined patterns, with a possible return to any previous steps
- Using this knowledge for promoting the performance of the system and resolving any potential conflicts with previously held beliefs or extracted knowledge

These are the steps that all KDD and data mining tasks progress through.

1.2.2 Data Mining Methods

The objective of data mining is both prediction and description. That is, to *predict* unknown or future values of the attributes of interest using other attributes in the databases, while *describing* the data in a manner understandable and interpretable to humans. Predicting the sale amounts of a new product based on advertising expenditure, or predicting wind velocities as a function of temperature, humidity, air pressure, etc., are examples of tasks with a predictive goal in data mining. Describing the different terrain groupings that emerge in a sampling of satellite imagery is an example of a descriptive goal for a data mining task. The relative importance of description and prediction can vary between different applications. These two goals can be fulfilled by any of a number data mining tasks including: classification, regression, clustering, summarization, dependency modeling, and deviation detection. (Harrell and Frank, 2001; Montgomery et al., 2001)

1.2.3 Predictive tasks

The following are general tasks that serve predictive data mining goals:

- *Classification* – to segregate items into several predefined classes. Given a collection of training samples, this type of task can be designed to find a model for class attributes as a function of the values of other attributes (Duda et al., 2001).
- *Regression* – to predict a value of a given continuously valued variable based on the values of other variables, assuming either a linear or nonlinear model of dependency. These tasks are studied in statistics and neural network fields (Montgomery et al., 2001).
- *Deviation Detection* – to discover the most significant changes in data from previously measured or normative values (Arning et al., 1996; Fayyad et al., 1996). Explicit information outside the data, like integrity constraints or predefined patterns, is used for deviation detection. Arning et al., (1996) approached the problem from the inside of the data, using the implicit redundancy.

1.2.4 Descriptive tasks

- *Clustering* – to identify a set of categories, or clusters, that describe the data (Jain & Dubes, 1988).
- *Summarization* – to find a concise description for a subset of data. Tabulating the mean and standard deviations for all fields is a simple example of

summarization. There are more sophisticated techniques for summarization and they are usually applied to facilitate automated report generation and interactive data analysis (Fayyad et al., 1996).

- *Dependency modeling* – to find a model that describes significant dependencies between variables. For example, probabilistic dependency networks use conditional independence to specify the *structural* level of the model and probabilities or correlation to specify the strengths (*quantitative* level) of dependencies (Heckerman, 1996).

1.2.5 Mixed tasks

There are some tasks in data mining that have both descriptive and predictive aspects. Using these tasks, we can move from basic descriptive tasks toward higher-order predictive tasks. Here, we indicate two of them:

- *Association Rule Discovery* – Given a set of records each of which contain some number of items from a given collection, produce dependency rules which will predict the occurrence of an item based on patterns found in the data.
- *Sequential Pattern Discovery* – Given a set of objects, where each object is associated with its own timeline of events, find rules that predict strong sequential dependencies among different events. Rules are formed by first discovering patterns followed by event occurrences which are governed by timing constraints found within those patterns.

So far we briefly described the main concepts of data mining. Chapter two focuses on methods and algorithms of data mining in the context of descriptive and predictive tasks. The research background of both the association rule and sequential pattern mining – newer techniques in data mining, that deserve a separate discussion – will be discussed in chapter five.

Data mining does not take place in a vacuum. In other words, any application of this method of analysis is dependent upon the context in which it takes place. Therefore, it is necessary to know the environment in which we are going to use data mining methods. The next section provides a brief overview of the LON-CAPA system.

1.3 Online Education systems

Several Online Education systems[1] such as Blackboard, WebCT, Virtual University (VU), and some other similar systems have been developed to focus on course management issues. The objectives of these systems are to present courses and instructional programs through the web and other technologically enhanced media. These new technologies make it possible to offer instruction without the limitations of time and place found in traditional university programs. However, these systems tend to use existing materials and present them as a static package via the Internet. There is another approach, pursued in LON-CAPA, to construct more-or-less new courses using newer network technology. In this model of content creation, college faculty, K-12 teachers, and students interested in collaboration can access a database of hypermedia software

[1] See http://www.edutools.info for an overview of current web-based educational systems.

modules that can be linked and combined (Kortemeyer and Bauer, 1999). The LON-CAPA system is the primary focus of this chapter.

1.3.1 LON-CAPA, System Overview

LON-CAPA is a *distributed* instructional management system, which provides students with personalized problem sets, quizzes, and exams. *Personalized (or individualized) homework* means that each student sees a slightly different computer-generated problem. LON-CAPA provides students and instructors with immediate feedback on conceptual understanding and correctness of solutions. It also provides faculty the ability to augment their courses with individualized, relevant exercises, and develop and share modular online resources. LON-CAPA aims to put this functionality on a homogeneously distributed platform for creating, sharing, and delivering course content with emphasis on cross-institutional collaboration and intellectual property rights management.

1.3.2 LON-CAPA Topology

LON-CAPA is physically built as a geographically distributed network of constantly connected servers. Figure 1.2 shows an overview of this network. All machines in the network are connected with each other through two-way persistent TCP/IP connections. The *network* has two classes of servers: library servers and access servers. A *library server* can act as a *home server* that stores all personal records of users, and is responsible for the initial authentication of users when a session is opened on any server in the network. For authors, it also hosts their construction area and the authoritative copy of every resource that has been published by that author. An *Access Server* is a machine that hosts student sessions. Library servers can be used as backups to host sessions when all access servers in the network are overloaded.

Every user in LON-CAPA is a member of one domain. *Domains* could be defined by departmental or institutional boundaries like MSU, FSU, OHIOU, or the name of a publishing company. These domains can be used to limit the flow of personal user information across the network, set access privileges, and enforce royalty schemes. Thus, the student and course data are distributed amongst several repositories. Each user in the system has one library server, which is his/her home server. It stores the authoritative copy of all of their records.

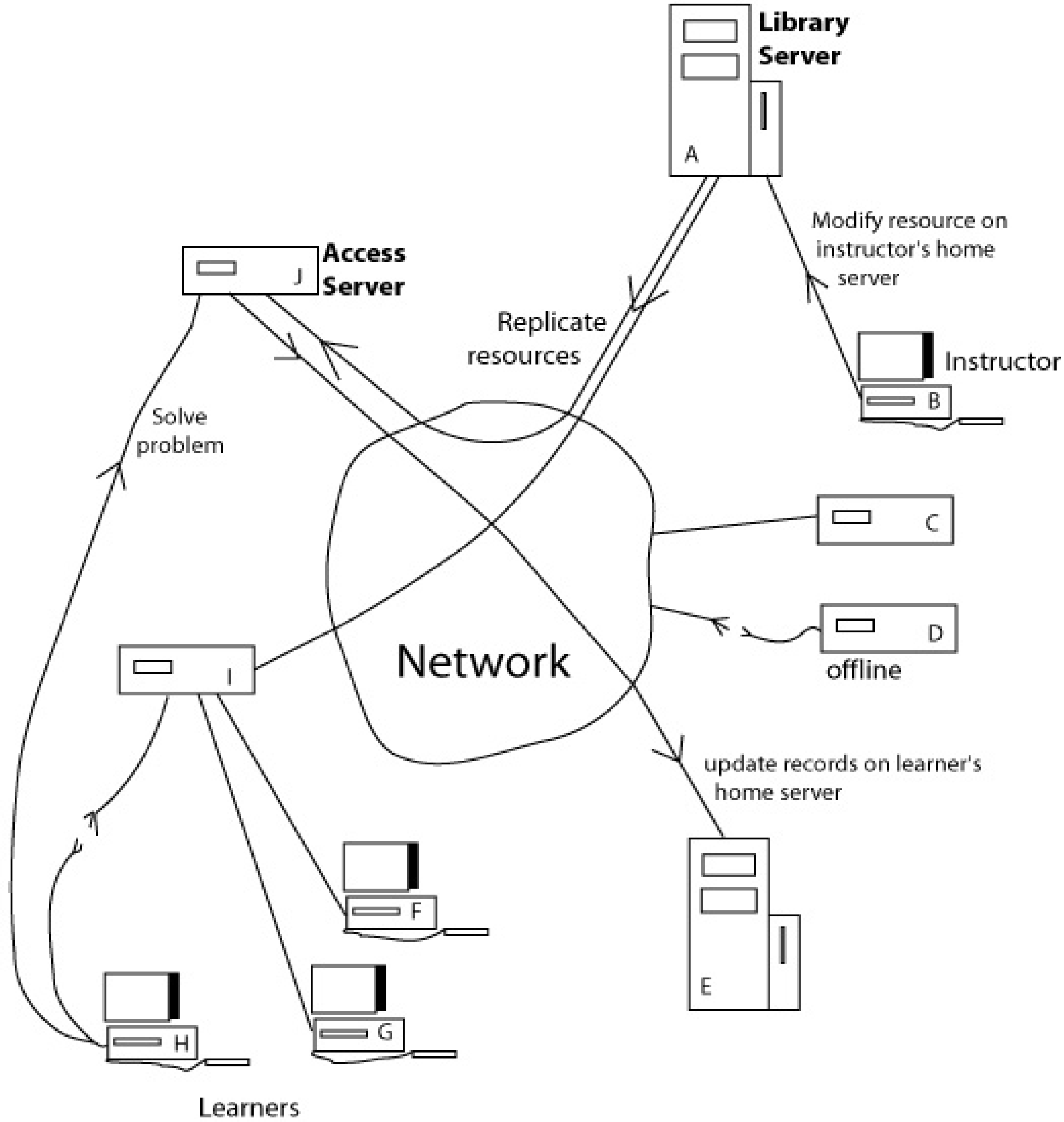


Figure 1.2 A schema of distributed data in LON-CAPA

LON-CAPA currently runs on Redhat-Linux Intel-compatible hardware. The current MSU production setup consists of several access servers and some library servers. All access servers are set up on a round-robin IP scheme as frontline machines, and are accessed by the students for “user session.” The current implementation of LON-CAPA uses mod_perl inside of the Apache web server software.

1.3.3 Data Distribution in LON-CAPA

Educational objects in LON-CAPA range from simple paragraphs of text, movies, and applets, to individualized homework problems. Online educational projects at MSU have produced extensive libraries of resources across disciplines. By combining these resources, LON-CAPA produces a national *distributed digital library* with mechanisms to store and retrieve these objects. Participants in LON-CAPA can publish their own objects in the common pool. LON-CAPA will allow groups of organizations (departments, universities, schools, commercial businesses) to link their online instructional resources in a common marketplace, thus creating an online economy for instructional resources (lon-capa.org). Internally, all resources are identified primarily by their URL.

LON-CAPA does enable faculty to combine and sequence these learning objects at several levels. For example, an instructor from Community College A in Texas can compose a page by combining a text paragraph from University B in Detroit with a movie from College C in California and an online homework problem from Publisher D in New York. Another instructor from High School E in Canada might take that page from Community College A and combine it with other pages into a module, unit or section. Those in turn can be combined into whole course packs.

1.3.4 Resource Variation in LON-CAPA

LON-CAPA provides three types of resources for organizing a course. LON-CAPA refers to these resources as *Content Pages*, *Problems*, and *Maps*. Maps may be either of two types: *Sequences* or *Pages*. LON-CAPA resources may be used to build the outline, or structure, for the presentation of the course to the students.

- A **Content Page** displays course content. It is essentially a conventional html page. These resources use the extension ".html".
- A **Problem** resource represents problems for the students to solve, with answers stored in the system. These resources are stored in files that must use the extension ".problem".
- A **Page** is a type of **Map** which is used to join other resources together into one HTML page. For example, a page of problems will appear as a problem set. These resources are stored in files that must use the extension ".page".
- A **Sequence** is a type of **Map**, which is used to link other resources together. Sequences are stored in files that must use the extension ".sequence". Sequences can contain other sequences and pages.

Authors create these resources and publish them in library servers. Then, instructors use these resources in online courses. The LON-CAPA system logs any access to these resources as well as the sequence and frequency of access in relation to the successful completion of any assignment. All these accesses are logged.

1.3.5 LON-CAPA Strategy for Data Storage

Internally, the student data is stored in a directory:

/home/httpd/lonUsers/domain/1st.char/2nd.char/3rd.char/username/

For example /home/httpd/lonUsers/msu/m/i/n/minaeibi/

Figure 1.3 shows a list of a student's data. Files ending with `.db` are `GDBM` files (Berkeley database), while those with a course-ID as name, for example `msu_12679c3ed543a25msul1.db`, store performance data for that student in the course.

```
ls -alF /home/httpd/lonUsers/msu/m/i/n/minaeibi

-rw-r--r--   1 www      users       13006 May 15 12:21 activity.log
-rw-r-----   1 www      users       12413 Oct 26  2000 coursedescriptions.db
-rw-r--r--   1 www      users       11361 Oct 26  2000 coursedescriptions.hist
-rw-r-----   1 www      users       13576 Apr 19 17:45 critical.db
-rw-r--r--   1 www      users        1302 Apr 19 17:45 critical.hist
-rw-r-----   1 www      users       13512 Apr 19 17:45 email_status.db
-rw-r--r--   1 www      users        1496 Apr 19 17:45 email_status.hist
-rw-r--r--   1 www      users       12373 Apr 19 17:45 environment.db
-rw-r--r--   1 www      users         169 Apr 19 17:45 environment.hist
-rw-r-----   1 www      users       12315 Oct 25  2000 junk.db
-rw-r--r--   1 www      users        1590 Nov  4  1999 junk.hist
-rw-r-----   1 www      users       23626 Apr 19 17:45 msu_12679c3ed543a25msul1.db
-rw-r--r--   1 www      users        3363 Apr 19 17:45 msu_12679c3ed543a25msul1.hist
-rw-r-----   1 www      users       18497 Dec 21 11:25 msu_1827338c7d339b4msul1.db
-rw-r--r--   1 www      users        3801 Dec 21 11:25 msu_1827338c7d339b4msul1.hist
-rw-r-----   1 www      users       12470 Apr 19 17:45 nohist_annotations.db
-rw-r-----   1 www      users      765954 Apr 19 17:45 nohist_email.db
-rw-r--r--   1 www      users      710631 Apr 19 17:45 nohist_email.hist
-rw-r--r--   1 www      users          13 Apr 19 17:45 passwd
-rw-r--r--   1 www      users       12802 May  3 13:08 roles.db
-rw-r--r--   1 www      users        1316 Apr 12 16:05 roles.hist
```

Figure 1.3 Directory listing of user's home directory

Courses are assigned to users, not vice versa. Internally, courses are handled like users without login privileges. The username is a unique ID, for example msu_12679c3ed543a25msul1 – every course in every semester has a unique ID, and there is no semester transition. The user-data of the course includes the full name of the course, a pointer to its top-level resource map ("course map"), and any associated deadlines, spreadsheets, etc., as well as a course enrollment list. The latter is somewhat redundant, since in principle, this list could be produced by going through the roles of all users, and looking for the valid role for a student in that course.

```
ls -alF /home/httpd/lonUsers/msu/1/2/6/12679c3ed543a25msul1/

-rw-r-----   1 www      users       17155 Apr 25 16:20 classlist.db
-rw-r--r--   1 www      users       60912 Apr 25 16:20 classlist.hist
-rw-r-----   1 www      users       12354 Jan  4 16:40 environment.db
-rw-r--r--   1 www      users          82 Jan  4 16:40 environment.hist
-rw-r-----   1 www      users      103030 May 15 14:47 nohist_calculatedsheets.db
-rw-r-----   1 www      users       13050 May  9 21:04 nohist_expirationdates.db
-rw-r--r--   1 www      users           6 Jan  4 16:40 passwd
-rw-r-----   1 www      users       17457 May  9 21:04 resourcedata.db
-rw-r--r--   1 www      users        8888 May  9 21:04 resourcedata.hist
```

Figure 1.4 Directory listing of course's home directory

An example of course data is shown in Figure 1.4. `classlist` is the list of students in the course, `environment` includes the course's full name, etc, and `resourcedata` are deadlines, etc. The parameters for homework problems are stored in these files.

To identify a specific instance of a resource, LON-CAPA uses symbols or "symbs." These identifiers are built from the URL of the map, the resource number of the resource in the map, and the URL of the resource itself. The latter is somewhat redundant, but might help if maps change. An example is

```
msu/korte/parts/part1.sequence___19___msu/korte/tests/part12.problem
```

The respective map entry is

```
<resource id="19" src="/res/msu/korte/tests/part12.problem"
 title="Problem 2">
</resource>
```

Symbs are used by the random number generator, as well as to store and restore data specific to a certain instance of a problem. More details of the stored data and their exact structures will be explained in chapter three, when we will describe the data acquisition of the system.

1.3.6 Resource Evaluation in LON-CAPA

One of the most challenging aspects of the system is to provide instructors with information concerning the quality and effectiveness of the various materials in the resource pool on student understanding of concepts. These materials can include web pages, demonstrations, simulations, and individualized problems designed for use on homework assignments, quizzes, and examinations. The system generates a range of statistics that can be useful in evaluating the degree to which individual problems are effective in promoting formative learning for students. For example, each exam problem contains attached metadata that catalog its degree of difficulty and discrimination for students at different phases in their education (i.e., introductory college courses, advanced college courses, and so on). To evaluate resource pool materials, a standardized format is required so that materials from different sources can be compared. This helps resource users to select the most effective materials available.

LON-CAPA has also provided questionnaires which are completed by faculty and students who use the educational materials to assess the quality and efficacy of resources. In addition to providing the questionnaires and using the statistical reports, we investigate here methods to find criteria for classifying students and grouping problems by examining logged data such as: time spent on a particular resource, resources visited (other web pages), due date for each homework, the difficulty of problems (observed statistically) and others. Thus, floods of data on individual usage patterns need to be gathered and sorted – especially as students go through multiple steps to solve problems, and choose between multiple representations of the same educational objects like video lecture demonstrations, a derivation, a worked example, case-studies, and etc. As the resource pool grows, multiple content representations will be available for use by students.

There has been an increasing demand for automated methods of resource evaluation. One such method is data mining, which is the focus of this research. Since the LON-CAPA data analyses are specific to the field of education, it is important to recognize the general context of using artificial intelligence in education.

The following section presents a brief review of intelligent tutoring systems – one typical application of artificial intelligence in the field of education. Note that herein the purpose is not to develop an intelligent tutoring system; instead we apply the main ideas of intelligent tutoring systems in an online environment, and implement data mining methods to improve the performance of the educational web-based system, LON-CAPA.

1.4 Intelligent Tutoring Systems (ITSs)

Intelligent tutoring systems are computer-based instructional systems that attempt to determine information about a student's learning status, and use that information to dynamically adapt the instruction to fit the student's needs. Examples of educational researchers who have investigated this area of inquiry are numerous: Urban–Lurain, 1996; Petrushin, 1995; Benyon and Murray, 1993; Winkkels, 1992; Farr and Psotka, 1992; Venezky and Osin, 1991; Larkin and Cabay, 1991; Goodyear, 1990; Frasson and Gauthier, 1988; Wenger, 1987; Yazdani, 1987. ITSs are often known as knowledge-based tutors, because they have separate knowledge bases for different domain knowledge. The knowledge bases specify what to teach and different instructional strategies specify how to teach (Murray, 1996).

One of the fundamental assumptions in ITS design is from an important experiment (Bloom, 1956) in learning theory and cognitive psychology, which states that individualized instruction is far superior to class-room style learning. Both the content and style of instruction can be continuously adapted to best meet the needs of a student (Bloom, 1984). Educational psychologists report that students learn best “by doing”, learn through their mistakes, and learn by constructing knowledge in a very individualized way (Kafaei and Resnik, 1996; Ginsburg and Opper, 1979; Bruner, 1966). For many years, researchers have argued that individualized learning offers the most effective and cognitively efficient learning for most students (Juel, 1996; Woolf, 1987).

Intelligent tutoring systems epitomize the principle of individualized instruction. Previous studies have found that intelligent tutoring systems can be highly effective learning aids (Shute and Regine, 1990). Shute (1991) evaluates several intelligent tutoring systems to judge how they live up to the main promise of providing more effective and efficient learning in relation to traditional instructional techniques. Results of such studies show that ITSs do accelerate learning.

1.4.1 Learning Enhancement in ITSs

In one study, Bloom (1984) states that conventional teaching methods provide the least effective method of learning. As instruction becomes more focused and individualized, learning is enhanced. He compares students' scores on achievement tests using three forms of instruction: conventional teaching, mastery teaching, and individualized tutoring. Mastery teaching is an instructional technique whereby a teacher supplements a lecture with diagnostic tests to determine where students are having problems, and adjusts the instruction accordingly. The results of this comparison are shown in Figure 1.5. Students receiving conventional teaching scored in the 50th percentile, students receiving mastery teaching scored in the 84th percentile, while students receiving individualized tutoring scored in the 98th percentile.

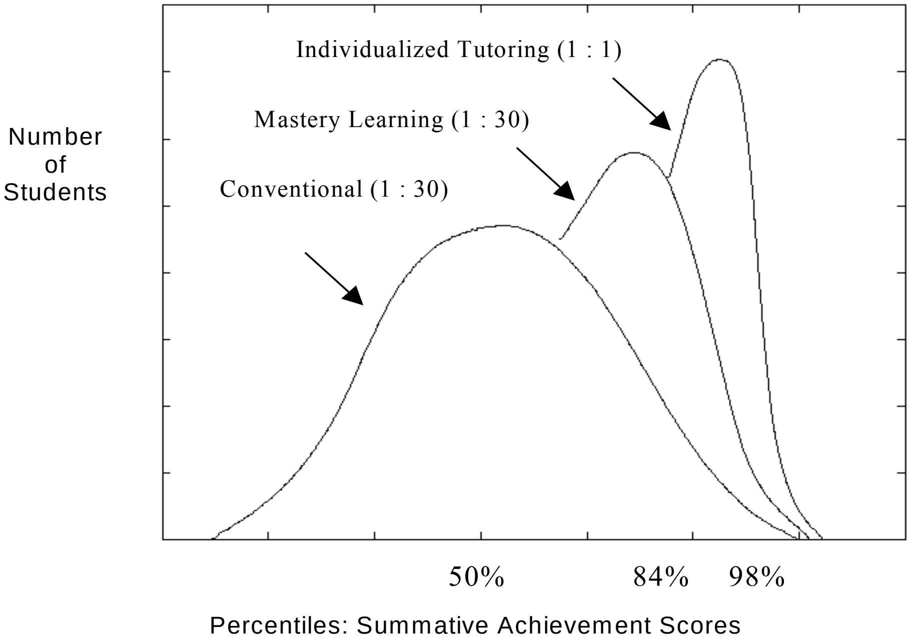


Figure 1.5 Distributions for different learning conditions (Adapted from Bloom, 1984)

Bloom replicates these results four times with three different age groups for two different domains, and thus, provides concrete evidence that tutoring is one of the most effective educational delivery methods available.

Since ITSs attempt to provide more effective learning through individualized instruction, many computer-assisted instruction techniques exist that can present instruction and interact with students in a tutor-like fashion, individually or in small groups. The incorporation of artificial intelligence techniques and expert systems technology to computer-assisted instruction systems gave rise to intelligent tutoring systems – i.e., systems that model the learner's understanding of a topic and adapt instruction accordingly. A few examples of systematically controlled evaluations of ITSs reported in the literature are shown in Table 1.1.

Table 1.1 Different Specific ITSs and their affects on learning rate

ITS	Literature	Objective	progress
LISP tutor	(Anderson, 1990)	Instructing LISP programming	1/3-2/3 less time
Smithtown	(Shute and Glaser, 1990)	Teach scientific inquiry skills	1/2 time, same knowledge
Sherlock	(Lesgold et al.,1990)	Avionics troubleshooting	1/5 time, same knowledge
Pascal ITS	(Shute, 1991)	Teach Pascal programming	1/3 time, same knowledge
Stat Lady	(Shute et al., 1993)	Instruct statistical procedures	More performance
Geometry Tutor	(Anderson et al., 1985)	Teach geometry theorems	Better solving

Shute and Poksta (1996) examine the results of these evaluations, which show that the tutors do accelerate learning with no degradation in outcome performance. The tutors should be evaluated with respect to the promises of ITSs – speed and effectiveness. In all cases, individuals using ITSs learned faster, and performed at least as well as those learning from traditional teaching environments. The results show that these individualized tutors could not only reduce the variance of outcome scores, but also increase the mean outcome dramatically.

1.4.2 Basic Architecture of an ITS

There is no standard architecture for an ITS. Nevertheless, four components emerge from the literature as part of an ITS (Wasson, 1997; Costa, 1992; Polson and Richardson, 1988; Yazdani, 1987; Wenger, 1987; Sleeman and Brown, 1982). These are the student model, the pedagogical module, the expert model, and the communication module or interface. These four components and their interactions are illustrated in Figure 1.6.

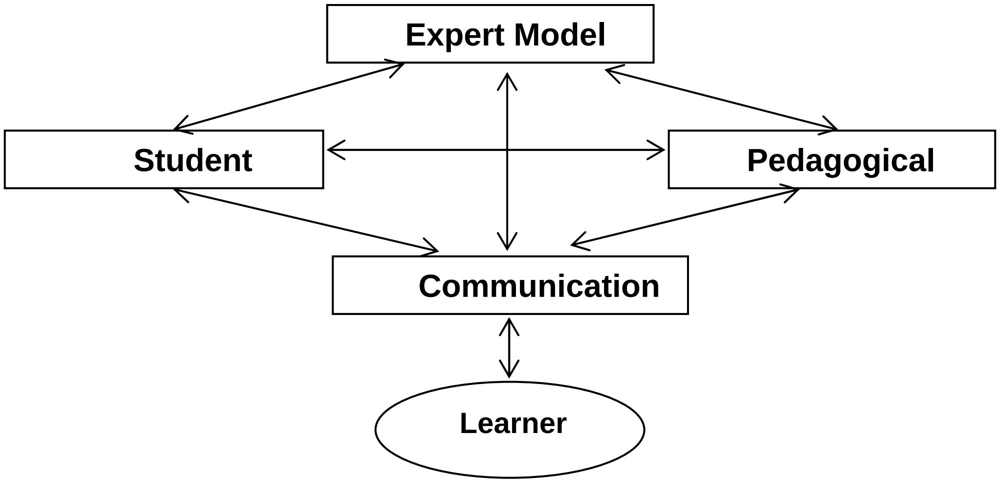


Figure 1.6 Components of an Intelligent Tutoring System (ITS)

The *student model* stores information of each individual learner. For example, such a model tracks how well a student is performing on the material being taught or records incorrect responses that may indicate misconceptions. Since the purpose of the student model is to provide data for the pedagogical module of the system, all of the information gathered should be usable by the tutor.

The pedagogical *module* provides a model of the teaching process. For example, information about when to review, when to present a new topic, and which topic to present is controlled by this module. As mentioned earlier, the student model is used as input to this component, so the pedagogical decisions reflect the differing needs of each student.

The *expert model* contains the domain knowledge, which is the information being taught to the learner. However, it is more than just a representation of the data; it is a model of how someone skilled in a particular domain represents the knowledge. By using an expert model, the tutor can compare the learner's solution to the expert's solution, pinpointing the places where the learner has difficulties. This component contains information the tutor is teaching, and is the most important since without it, there would be nothing to teach the student. Generally, this aspect of ITS requires significant knowledge engineering to represent a domain so that other parts of the tutor can access it.

The communication module controls interactions with a student, including the dialogue and the screen layouts. For example, it determines how the material should be presented to the student in the most effective way.

These four components – the student model, the pedagogical module, the expert model, and the communication module – shared by all ITSs, interact to provide the individualized educational experience promised by technology. The orientation or structure of each of these modules, however, varies in form depending on the particular ITS.

Current research trends focus on making tutoring systems truly "intelligent," in the artificial sense of the word. The evolution of ITSs demands more controlled research in four areas of intelligence: the domain expert, student model, tutor, and interface.

- The domain knowledge must be understood by the computer well enough for the expert model to draw inferences or solve problems in the domain.
- The system must be able to deduce a student's approximation of that knowledge.
- The tutor must be intelligent to the point where it can reduce differences between the expert and student performance.
- The interface must possess intelligence in order to determine the most effective way to present information to the student.

For ITSs to have a great impact on education, these and other issues must be resolved. To take advantage of newer, more effective instructional techniques, ITSs of the future will have to allow for increased student initiative and inter-student collaboration (Shute and Psotka, 1996). ITSs must also assess learning as it transfers to authentic tasks, not standardized tests, and establish connections across fields so that topics are not learned in isolation. A more fruitful approach for ITS development may be to develop specific cognitive tools, for a given domain or applicable across domains.

Such a transition would allow future ITSs to be everywhere, as embedded assistants that explain, critique, provide online support, coach, and perform other ITS activities.

1.4.3 Learning and Cognition Issues for ITS Development and Use

There are some findings in the areas of cognition and learning processes that impact the development and use of intelligent tutoring systems. Many recent findings are paving the way towards improving our understanding of learners and learning (Bransford, Brown et al., 2000). Learners have preconceptions about how the world works. If their initial understanding is not referenced or activated during the learning process, they may fail to understand any new concepts or information.

One key finding regarding competence in a domain is the need to have a more than a deep knowledge base of information related to that domain. One must also be able to understand that knowledge within the context of a conceptual framework – the ability to organize that knowledge in a manner that facilitates its use. A key finding in the learning and transfer literature is that organizing information into a conceptual framework allows for greater transfer of knowledge. By developing a conceptual framework, students are able to apply their knowledge in new situations and to learn related information more quickly. For example, a student who has learned problem solving for one topic in the context of a conceptual framework will use that ability to guide the acquisition of new information for a different topic within the same framework. This fact is explained by Hume (1999): "When we have lived any time, and have been accustomed to the uniformity of nature, we acquire a general habit, by which we always transfer the known to the unknown, and conceive the latter to resemble the former."

A relationship exists between the learning and transfer of knowledge to new situations. Transferring is usually a function of the relationships between what is learned and what is tested. For students to transfer knowledge successfully across domains, they must conceive of their knowledge base in terms of continuous, rather than discrete steps.

Recent research by Singley and Anderson indicates that the transfer of knowledge between tasks is a function of the degree to which the tasks share cognitive elements (Singley and Anderson, 1989). In their study, Singley and Anderson taught students several text editors, one after the other. They found that students learned subsequent text editors more rapidly and that the number of procedural elements shared by the two text editors predicted the amount of transfer. Their results showed that there was large transfer across editors that were very different in surface structures but had common abstract structures. Singley and Anderson were able to generate similar results for the transfer of mathematical competence across multiple domains.

Emerging computer-based technologies hold great promise as a means of supporting and promoting learning. There are several ways that such technology can be used to help meet the challenges of developing effective learning environments (El-Sheikh, 2001):

- Bringing real-world problems to the learning environment.
- Providing "scaffolding" support to students during the learning process. Scaffolding allows students to participate in complex cognitive experiences, such as model-based learning, that is more difficult without technical support.
- Increasing opportunities for learners to receive feedback and guidance from software tutors and learning environments.

- Building local and global communities of teachers, administrators, students, and other interested learners.
- Expanding opportunities for teachers' learning.

Learning environments need to be developed and implemented with a full understanding of the principles of learning and developmental psychology. In addition, these new learning environments need to be assessed carefully, including how their use can facilitate learning, as well as the cognitive, social, and learning consequences of using these new tools.

1.5 Summary

This research addresses data mining methods for extracting useful and interesting knowledge from the large data sets of students using LON-CAPA educational resources. The purpose is to develop techniques that will provide information that can be usefully applied by instructors to increase student learning, detect anomalies in homework problems, design the curricula more effectively, predict the approaches that students will take for some types of problems, *and provide appropriate advising for students in a timely manne*r, etc. This introductory chapter provided an overview of the LON-CAPA system, the context in which we are going to use data mining methods. In addition, a brief introduction to Intelligent Tutoring Systems provided examples of expert systems and artificial intelligence in educational software. Following this, it is necessary to analyze data mining methods that can be applied within this context in greater detail.

Chapter 2 Background on Data Mining Methods

In the previous chapter we described the basic concepts of data mining. This chapter focuses on methods and algorithms in the context of descriptive and predictive tasks of data mining. We describe clustering methods in data mining, and follow this with a study of the classification methods developed in related research while extending them for predictive purposes. The research background for both association rule and sequential pattern mining will be presented in chapter five.

2.1 Classification and Predictive Modeling

Classification is used to find a model that segregates data into predefined classes. Thus classification is based on the features present in the data. The result is a description of the present data and a better understanding of each class in the database. Thus classification provides a model for describing future data (Duda et al., 2001; McLachlan, 1992; Weiss and Kulikowski, 1991; Hand, 1987). *Prediction* helps users make a decision. Predictive modeling for knowledge discovery in databases predicts unknown or future values of some attributes of interest based on the values of other attributes in a database (Masand and Shapiro, 1996). Different methodologies have been used for classification and developing predictive modeling including Bayesian inference (Kontkanen et al., 1996), neural net approaches (Lange, 1996), decision tree-based methods (Quinlan, 1986) and genetic algorithms-based approaches (Punch et al., 1995).

2.1.1 Bayesian Classifier

One of the major statistical methods in data mining is *Bayesian inference*. The naive Bayesian classifier provides a simple and effective approach to classifier learning. It assumes that all class-conditional probability densities are completely specified. Even though this assumption is often violated in real world data sets, a *naïve* Bayesian classifier (where a small number of parameters are not completely specified) is employed (Jain et al., 2000; Duda et al., 2001; Wu et al., 1991). The Bayes classifier shown in Figure 2.1 can be explained as follows: A set of patterns $a_j, j = 1,...,n$, is given, and every pattern is sensed by a sensing device which is capable of capturing the features. Each pattern is considered in terms of a measurement vector x_i. A pattern a_j belongs to a classification set ω_i, which includes all the possible classes that can be assigned to pattern a_j. For the sake of simplicity, all feature measurements are considered identical and each pattern belongs only to one of the m-possible classes ω_i, $i = 1,...,m$.

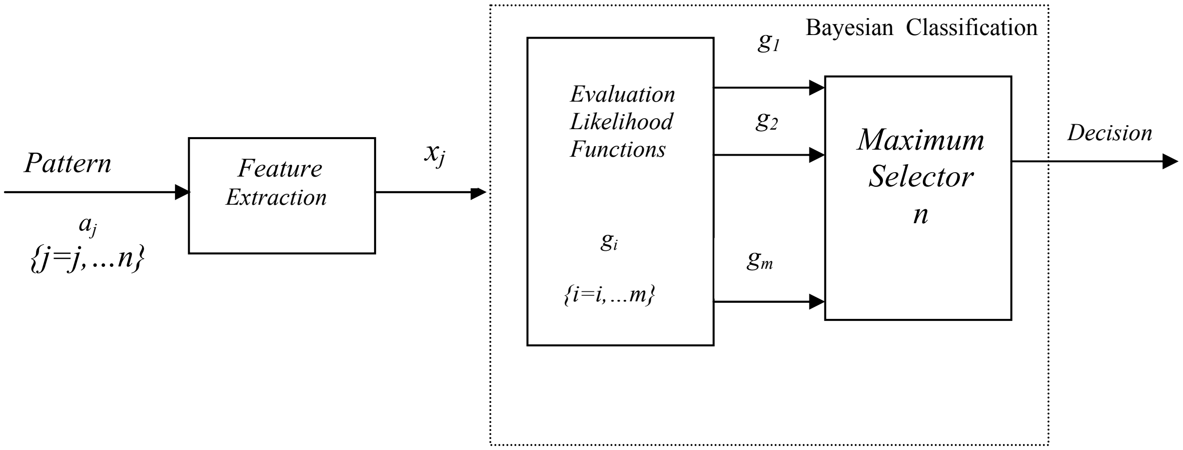


Figure 2.1 The Bayesian Classification Process (Adapted from Wu et al., 1991)

To classify a pattern into one of the m classes, a feature space is constructed according to the measurement vector x, which is considered to be a measurement of true values damaged by random noisy data. The class-conditional probability density functions estimated from training data represent the uncertainty in discovered knowledge.

$$p(x \mid \omega_i)\,,\ i = 1,\ldots,m. \tag{2.1}$$

Bayes decision theory states that the *a-posteriori* probability that an event may be calculated according to the following equation:

$$p(\omega_i \mid x) = \frac{p(x \mid \omega_i)\,p(\omega_i)}{p(x)}\ ,\ i = 1,\ldots,m. \tag{2.2}$$

Eventually, the decision criteria can be applied for classification. To gain the optimal solution, the maximum likelihood classification or the Bayesian minimum error decision rule is applied. It is obtained by minimizing the misclassification and errors in classification. Thus, a pattern is classified into class ω_i with the highest *posteriori* probability or likelihood:

$$g_i = \max{}_j \{g_i\}, j = 1,\ldots,m. \tag{2.3}$$

The quadratic discriminant function using the Bayesian approach is the most common method in supervised parametric classifiers. If the feature vectors are assumed to be Gaussian in distribution, the parameters of the Gaussians are estimated using maximum likelihood estimations. The discriminant function decision rule and the *a-posteriori* probabilities for each classification are calculated for each sample test, *x*, using the following equation (Duda et al., 2001):

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T \textstyle\sum_i^{-1}(x-\mu_i) - \frac{1}{2}\ln|\textstyle\sum_i| + \ln p(\omega_i)\ , \tag{2.4}$$

where x is a d×1 vector representing any point in the d-dimensional feature space, μ_i (also a $d \times 1$ vector) is the sample mean of the ith class training sample, and $\sum_i (d \times d)$ is the sample covariance matrix of the ith class training sample. To obtain the optimal solution, the maximum likelihood classification or the Bayesian minimum error decision rule is applied. The sample is then assigned to the class that produces the highest *a-posteriori* probability. It is obtained by minimizing the misclassification and errors in classification.

2.1.2 Decision tree-based method

Decision tree-based methods are popular methods for use in a data mining context. The decision tree classifier uses a hierarchical or layered approach to classification. Each vertex in the tree represents a single test or decision. The outgoing edges of a vertex correspond to all possible outcomes of the test at that vertex. These outcomes partition the set of data into several subsets, which are identified by every leaf in the tree. A leaf of the tree specifies the expected value of the categorical attribute for the records described by the path from the root to that leaf. Learned trees can also be represented as sets of if-then-else rules. (Mitchell, 1997)

An instance is classified by starting at the root node of the tree. At each level of the tree the attributes of an instance are matched to a number of mutually exclusive nodes. The leaf nodes assign an instance to a class. The classification of an instance therefore involves a sequence of tests where each successive test narrows the interpretation. The sequence of tests for the classifier is determined during a training period. Given some new data, the ideal solution would test all possible sequences of actions on the attributes of the new data in order to find the sequence resulting in the minimum number of misclassifications.

Tree-based classifiers have an important role in pattern recognition research because they are particularly useful with non-metric data (Duda et al., 2001). Decision tree methods are robust to errors, including both errors in classifying the training examples and errors in the attribute values that describe these examples. Decision tree can be used when the data contain missing attribute values. (Mitchell, 1997)

Most algorithms that have been developed for decision trees are based on a core algorithm that uses a top-down, recursive, greedy search on the space of all possible decision trees. This approach is implemented by ID3 algorithm2 (Quinlan, 1986) and its successor C4.5 (Quinlan, 1993). C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, and rule derivation. The rest of this section discusses some important issues in decision trees classifiers.

2.1.2.1 What is the best feature for splitting?

The first question that arises in all tree-based algorithms concerns which properties are tested in each node? In other words, which attribute is the “most informative” for the classifier? We would like to select the attribute that is the most informative of the attributes not yet considered in the path from the root. This establishes what a "Good" decision tree is. Entropy is used to measure a node’s information. Claude Shannon (1984) introduced this notion in “Information Theory”. Based on entropy, a statistical property

[2] ID3 got this name because it was the third version of “interactive dichotomizer” procedure.

called *information gain* measures how well a given attribute separates the training examples in relation to their target classes.

2.1.2.1.1 Entropy impurity

Entropy characterizes the *impurity* of an arbitrary collection of examples *S* at a specific node *N*. Sometimes (Duda et al., 2001) the impurity of a node *N* is denoted by *i(N)*.

$$Entroy(S) = i(N) = -\sum_{j} P(\omega_j)\log_2 P(\omega_j) \qquad (2.5)$$

where $P(\omega_j)$ is the fraction of examples at node *N* that go to category ω_j.

If all the patterns are from the same category the impurity is 0, otherwise it is positive; if all categories are equally distributed at node *N* then the impurity has its greatest value 1.

The key question then is, on the decision path from the root to node *N*, what features and their values should we select for the test at node *N* when property query *T* is used? A heuristic is suggested to select a query that decreases the impurity as much as possible (Duda et al., 2001).

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R) \qquad (2.6)$$

where N_L and N_R are the left and right descendent nodes, and the $i(N_L)$ and $i(N_R)$ are their impurities respectively, and P_L is fraction of patterns at node N that will go to N_L when the property query T is used. The goal of the heuristic is to maximize Δi, thus minimizing the impurities corresponds to an information gain which is provided by the query.

2.1.2.1.2 Gini impurity

One can rank and order each splitting rule on the basis of the quality-of-split criterion. Gini is the default rule in CART because it is often the most efficient splitting rule. Essentially, it measures how well the splitting rule separates the classes contained in the parent node (Duda et al., 2001).

$$i(N) = \sum_{j \neq i} P(\omega_j)P(\omega_i) = 1 - \sum_{j} P^2(\omega_j) \tag{2.7}$$

As shown in the equation, it is strongly peaked when probabilities are equal. So what is Gini trying to do? Gini attempts to separate classes by focusing on one class at a time. It will always favor working on the largest or, if you use costs or weights, the most "important" class in a node.

2.1.2.1.3 Twoing impurity

An alternative criterion also available in CART is Twoing impurity. The philosophy of Twoing is different from that of Gini. Rather than initially pulling out a single class, Twoing first segments the classes into two groups, attempting to find groups that together add up to 50 percent of the data. Twoing then searches for a split to separate the two subgroups (Duda et al., 2001). This is an ideal split. It is unlikely that any real-world database would allow you to cleanly separate four important classes into two subgroups in this way. However, splits that approach this ideal might be possible, and these are the splits that Twoing seeks to find.

2.1.2.2 How to Avoid Overfitting

If we continue to grow the tree until each leaf node corresponds to the lowest impurity then the data is typically overfit. In some cases every node corresponds to a single training input. In such cases we cannot expect an appropriate generalization in noisy problems having high Bayes error (Duda et al. 2001; Russell and Norvig, 1997). On the other hand if we stop splitting early, then a high performance tree classifier will not be achieved. There are several approaches to avoid overfitting in the training phase of tree-based classification:

2.1.2.2.1 Cross-Validation

Cross-validation is a technique to eliminate the occurrence of overfitting. The main idea of cross-validation is to estimate how well the current hypothesis will predict unseen data (Duda et al. 2001; Russell and Norvig, 1997). This is done by randomly dividing the data into two subsets, training and test. Usually, the test subset is a fraction of all of the data, i.e., 10%. The hypothesis induced from the training phase is tested on the rest of data to get the prediction performance. This should be repeated on different subsets of data, and then the result averaged. Cross-validation should be used in order to select a tree with good prediction performance.

2.1.2.2.2 Setting a threshold

Another method for overfitting avoidance is to consider a small threshold value in minimizing the impurity. We stop splitting when the impurity at a node is reduced by less than the considered threshold. The benefit of this method over the cross-validation is that the tree is trained using all the training data. Another benefit of this method is that leaf nodes can lie at different levels of the tree.

2.1.2.2.3 Pruning

The principal alternative of stop-splitting is pruning (Duda et al., 2001). One approach, called *reduced-error pruning* (Quinlan, 1987), sets each node in the decision tree to be candidate for pruning. "Pruning a decision node consists of removing the subtree rooted at that node, making it a leaf node, and assigning it the most common classification of the training examples affiliated with that node. Nodes are removed only if the resulting pruned tree performs no worse than the original over the validation set." (Mitchell, 1997)

In C4.5, Quinlan (1993) applied a successful technique for finding high accuracy hypotheses during the pruning process, which is called *rule post pruning*. It involves the following steps:

1. Induce the decision tree from the training set, growing the tree until the training data is fully fitted as well as possible, allowing overfitting to occur.
2. Convert the learned tree into an equivalent set of rules by creating a rule corresponding to a path from the root to a leaf node.
3. Prune each rule by deleting any preconditions that lead to promoting the estimated accuracy.
4. Sort the pruned rules by their estimated accuracy, and set them in a sequence for classifying the instances.

Why should we prefer a short hypothesis? We wish to create small decision trees so that records can be identified after only a few questions. According to Occam's Razor, "*Prefer the simplest hypothesis that fits the data*" (Duda et al., 2001).

2.1.2.3 Drawbacks of decision tree

The drawback of decision trees is that they are implicitly limited to talking about a single data point. One cannot consider two or more objects to make a decision about, thus the decision boundary in this classifier is linear and often too coarse. In other words, once a node is split, the elements in a child stay in that sub-tree forever. Therefore, the decision tree classifier often yields a sub-optimal decision boundary. Another drawback of decision trees is that they are very instable in the presence of noisy data (Duda et al., 2001; Mitchell, 1997).

2.1.3 Neural Network Approach

A *neuron* is a special biological cell with information processing ability (Jain et al., 1996). The classification approach based on an Artificial Neural Network (ANN) (a connectionist model) generates a lower classification error rate than the decision tree approach in several cases, yet it requires more training time (Quinlan, 1994; Russle and Norvig, 1995). A neural network is usually a layered graph with the output of one node feeding into one or more nodes in the next layer.

The Multi-layer Perceptron (MLP) is a basic feedforward artificial neural network using a back-propagation algorithm for training. That is, during training, information is propagated back through the network and used to update connection weights. According to Ruck et al., (1990), *multi-layer perceptron* training uses the back-propagation learning algorithm to approximate the optimal discriminant function defined by Bayesian theory. The output of the *MLP* approximates the *posteriori* probability functions of the classes being trained. The Sigmoidial activation function is used for learning the input weight vectors in the training phase as follows:

$$f(x) = \frac{1}{1+e^{(-x)}} \tag{2.8}$$

Tuning each of the learning rate, the number of epochs, the number of hidden layers, and the number of neurons (nodes) in every hidden layer is a very difficult task and all must be set appropriately to reach a good performance for MLP. In each epoch the input data are used with the present weights to determine the errors, then back-propagated errors are computed and weights updated. A bias is provided for the hidden layers and output.

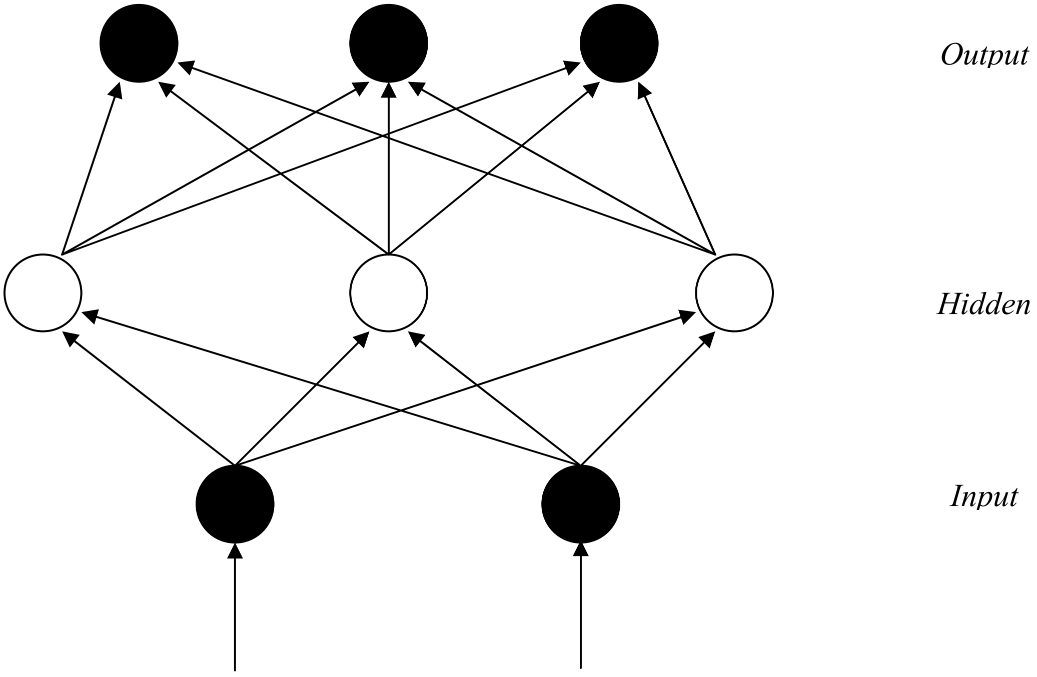


Figure 2.2 A Three Layer Feedforward Neural Network (Lu et al., 1995)

Adopting data mining techniques to MLP is not possible without representing the data in an explicit way. Lu et al.(1995) made an effort to overcome this obstacle by using a three-layer neural network to perform classification, which is the technique employed in this study. ANNs are made up of many simple computational elements (neurons), which are densely interconnected. Figure 2.2 shows a three-layer feedforward network, which has an input layer, a hidden layer, and an output layer. A node (neuron) in the network has a number of inputs and a single output. Every link in the network is associated with a weight. For example, node N_i has x_1^i, …, x_n^i as its inputs and a^i as its output. The input links of N_i have weights w_1^i, …, w_n^i. A node generates its output (the activation value) by summing up its input weights, subtracting a threshold and passing the result to a non-linear function f (activation function). Outputs from neurons in a layer are fed as inputs to next layer. Thus, when an input tuple (x_1, ..., x_n) is applied to the input layer of a network, an output tuple (c_1, ..., c_m) is obtained, where c_i has value 1 if the input tuple belongs to class c_i and 0 otherwise.

Lu et al.'s approach uses an ANN to mine classification rules through three steps explained as follows:

1. In the first step, a three-layer network is trained to find the best set of weights to classify the input data at a satisfactory level of accuracy. The initial weights are selected randomly from the interval [-1, 1]. These weights are then updated according to the gradient of the error function. This training phase is terminated when the norm of the gradient falls below a preset threshold.

2. Redundant links (paths) and nodes (neurons) that is, those nodes that don't have any effects on performance are removed and therefore, and a pruned network is obtained.
3. Comprehensible and concise classification rules are extracted from the pruned network in the form of: "if (a_1 θ v_1) & (a_2 θ v_2) & … & (a_n θ v_n) then C_j where an a_i is an input attribute value, v_i is a constant, θ is a relational operator (=, ≤, ≥, <,>), and C_j is one of the class labels.

2.1.4 *k*-Nearest Neighbor (*k*NN) Decision Rule

The k-nearest neighbor algorithm makes a classification for a given sample without making any assumptions about the distribution of the training and testing data. Each testing sample must be compared to all the samples in the training set in order to classify the sample. In order to make a decision using this algorithm, the distances between the testing sample and all the samples in the training set must first be calculated. In this proposal, the Euclidean distance is calculated, but, in general, any distance measurement may be used. The euclidean distance metric requires normalization of all features into the same range. At this point, the k closest neighbors of the given sample are determined where k represents an integer number between 1 and the total number of samples. The testing sample is then assigned to the label most frequently represented among the k nearest samples (Duda et al., 2001). The value of k that is chosen for this decision rule has an affect on the accuracy of the decision rule. The *k-nearest neighbor* classifier is a nonparametric classifier that is said to yield an efficient performance for optimal values of *k*.

2.1.5 Parzen Window classifier

In this approach a *d*-dimensional window is formed around all the training samples and then, based on the number of patterns that fit in those windows, the probability estimates of the different classes are made. This can be stated as follows (Duda et al., 2001):

$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{v_n}\varphi\frac{(x - x_i)}{h_n} \qquad (2.9)$$

where v_n is a d-dimensional hypercube in the feature space, and φ is a general probability distribution function. $p_n(x)$ is the probability that the pattern fits in the given class. It is necessary to choose the form of φ. One can assume a multivariate normal distribution for φ. The windows are centered on the training points, hence, the mean is known, but there is no predefined method to determine the variance. Depending upon the problem under study, the variance is estimated by minimizing the error rate and maximizing the classifier performance. Therefore, it needs to be determined by trial and error. In our experiment we assume that the classes are independent and thus the covariance matrices for the Gaussian distribution are diagonal.

2.2 Clustering

Data clustering is a sub-field of data mining dedicated to incorporating techniques for finding similar groups within a large database. *Data clustering* is a tool for exploring data and finding a valid and appropriate structure for grouping and classifying the data (Jain & Dubes, 1988). A *cluster* indicates a number of similar objects, such that the members inside a cluster are as similar as possible (homogeneity), while at the same time the objects within different clusters are as dissimilar as possible (heterogeneity) (Hoppner et al., 2000). The property of homogeneity is similar to the cohesion attribute between objects of a class in software engineering, while heterogeneity is similar to the coupling attribute between the objects of different classes.

Unlike data classification, data clustering does not require category labels or predefined group information. Thus, clustering has been studied in the field of machine learning as a type of *unsupervised learning,* because it relies on "learning from observation" instead of "learning from examples." The pattern proximity matrix could be measured by a distance function defined on any pairs of patterns (Jain & Dubes, 1988; Duda et al., 2001).). A simple distance measure i.e., Euclidean distance can be used to express dissimilarity between every two patterns.

The grouping step can be performed in a number of ways. "Hierarchical clustering algorithms produce a nest series of partitions based on a criterion for merging or splitting clusters based on similarity. Partitional clustering algorithms identify the partition that optimizes a clustering criterion" (Jain et al. 1999). Two general categories of clustering methods are partitioning method, and hierarchical method – both of which are employed in analysis of the LON-CAPA data sets.

2.2.1 Partitional Methods

A partitional algorithm assuming a set of *n* objects in *d*-dimensional space and an input parameter, *k*, organizes the objects into *k* clusters such that the total deviation of each object from its cluster center is minimized. The deviation of an object in a cluster depends on the *similarity* function, which in turn depends on the *criterion* employed to distinguish the objects of different clusters. Clusters can be of arbitrary shapes and sizes in multidimensional space. Every particular clustering criterion implies a specified structure for the data. These criteria are employed in some of the most popular partitioning methods: square error approach, mixture model, mode estimation, graph connectivity, and nearest neighbor relationship.

The most common approach in these methods is to optimize the criterion function using an iterative, hill-climbing technique. Starting from an initial partition, objects are moved from one cluster to another in an effort to improve the value of the criterion function (Jain & Dubes, 1988). Each algorithm has a different way for representing its clusters.

2.2.1.1 k-mean Algorithm

The *k*-means algorithm is the simplest and most commonly used clustering algorithm employing a square error criterion (McQueen 1967). It is computationally fast, and iteratively partitions a data set into *k* disjoint clusters, where the value of *k* is an algorithmic input (Jain & Dubes, 1988; Duda et al. 2001). The goal is to obtain the partition (usually of hyper-spherical shape) with the smallest square-error. Suppose *k* clusters {*C*$_1$, *C*$_2$, ..., *C*$_k$} such that *C*$_k$ has *n*$_k$ patterns. The mean vector or center of cluster *C*$_k$

$$\mu^{(k)} = \frac{1}{n_k}\sum_{i=1}^{n_k} x_i^{(k)} \tag{2.10}$$

where *n*$_i$ is number of patterns in cluster *C*$_i$, (among exactly *k* clusters: *C*$_1$, *C*$_2$, ..., *C*$_k$) and *x* is the point in space representing the given object.

The total squared-error: $E_K^2 = \sum_{k=1}^{K} e_k^2$ where $e_k^2 = \sum_{i=1}^{n_k} (x_i^{(k)} - \mu^k)^T \ (x_i^{(k)} - \mu^k)$ which is computed in this way:

The steps of the iterative algorithm for partitional clustering are as follows:

1. Choose an initial partition with k < n clusters (μ^1, μ^2 , … , μ^k) are cluster centers and n is the number of patterns).
2. Generate a new partition by assigning a pattern to its nearest cluster center μ^i.
3. Recompute new cluster centers μ^i.
4. Go to step 2 unless there is no change in μ^i.
5. Return μ^1, μ^2 , … , μ^k as the mean values of *C*$_1$, *C*$_2$, ..., *C*$_k$.

The idea behind this iterative process is to start from an initial partition assigning the patterns to clusters and to find a final partition containing *k* clusters that minimizes *E* for fixed *k*. In step 3 of this algorithm, *k*-means assigns each object to its nearest center forming a set of clusters. In step 4, all the centers of these new clusters are recomputed with function *E* by taking the mean value of all objects in every cluster. This iteration is repeated until the criterion function E no longer changes. The k-means algorithm is an efficient algorithm with the time complexity of $O(ndkr)$, where n is the total number of objects, d is the number of features, k is the number of clusters, and r is the number of iterations such that $r<k<n$.

The weaknesses of this algorithm include a requirement to specify the parameter k, the inability to find arbitrarily shaped clusters, and a high sensitivity to noise and outlier data. Because of this, Jain & Dubes, (1988) have added a step before step 5: "Adjust the number of clusters by merging and splitting the existing clusters or by removing small, or outlier clusters".

Fuzzy k-means clustering (soft clustering). In the k-means algorithm, each data point is allowed to be in exactly one cluster. In the fuzzy clustering algorithm we relax this condition and assume that each pattern has some "fuzzy" membership in a cluster. That is, each object is permitted to belong to more than one cluster with a graded membership. Fuzzy clustering has three main advantages: 1) it maps numeric values into more abstract measures (fuzzification); 2) student features (in LON-CAPA system) may *overlap* multiple abstract measures, and there may be a need to find a way to cluster under such circumstances; and 3) most real-world classes are fuzzy rather than crisp. Therefore, it is natural to consider the fuzzy set theory as a useful tool to deal with the classification problem (Dumitrescu et al., 2000).

Some of the fuzzy algorithms are modifications of the algorithms of the square error type such as k-means algorithm. The definition of the membership function is the most challenging point in a fuzzy algorithm. Baker (1978) has presented a membership function based on similarity decomposition. The similarity or affinity function can be based on the different concept such as Euclidean distance or probability. Baker and Jain (1981) define a membership function based on mean cluster vectors. Fuzzy partitional clustering has the same steps as the squared error algorithm, which is explained in the *k*-means algorithm section.

2.2.1.2 Graph Connectivity

A graph can represent the relationship between patterns. Every vertex represents a pattern and every edge represents the relation between two patterns. The edge weights are distances between two adjacent vertices. The criterion function here is that the pairs of patterns, which belong to the same cluster, should be closer than any pair belonging to different clusters. Several graph structures, such as minimum spanning trees (MST) (Zahn, 1971), relative neighborhood graphs (Toussaint, 1980), and Gabriel Graphs (Urquhart, 1982), have been applied to present a set of patterns in order to detect the clusters. For example, Zahn's algorithm consists of the following steps: 1) Create the MST for the entire set of *N* patterns. 2) Determine the inconsistent edges. 3) Delete the inconsistent edges from MST. 4) The remaining components are our clusters.

This algorithm can be applied recursively on the resulting components to determine new clusters. The heart of this algorithm lies in the definition of inconsistency. Zahn (1971) presents several criteria for inconsistency. An edge is inconsistent if its weight is much larger than the average of all other nearby edge weights. Other definitions of inconsistency are dependent on either the ratio between, or the standard deviation in which an edge weight differs from the average of nearby edge weights.

2.2.1.3 Nearest Neighbor Method

Since the distance matrix gives us an intuition of a cluster, nearest-neighbor distances can be used as the core of a clustering method. In this method every pattern is assigned to the same cluster as its nearest neighbor. An iterative procedure was proposed by Lu and Fu (1978) where each unlabeled pattern is assigned to the cluster of its nearest labeled neighbor pattern, provided the distance to that labeled neighbor is below a threshold. The process continues until all patterns are labeled or no additional labeling can occur. This algorithm partitions a set of $\{x_1, x_2, ..., x_n\}$ patterns into a set of *k* clusters $\{C_1, C_2, ..., C_k\}$. The user should specify a threshold *r,* which determines the maximum acceptable nearest neighbor distance. The steps of this algorithm are described in Jain & Dubes (1988). The number of clusters *k* which are generated is a function of the parameter *r*. As the value of r is increased the number of clusters *k* is decreased..

2.2.1.4 Mixture density clustering

The mixture resolving method assumes that the patterns are drawn from a particular distribution, and the goal is to identify the parameters of that distribution. Usually it is assumed that the individual components of the target partition are the mixture of a Gaussian distribution, and thus the parameters of the individual Gaussians are to be estimated (Jain et al., 1999). There are traditional approaches to use a maximum likelihood estimate of the parameter vectors for the component densities (Jain & Dubes, 1988). Dempster et al. (1977) proposed a general framework for maximum likelihood using the Expectation Maximization (EM) algorithm to estimate the parameters for missing data problems. The EM algorithm widely employed to estimate maximum likelihood estimation in in-complete data problems where there are missing data (McLachlan & Krishnan, 1997).

The EM algorithm is an iterative method for learning a probabilistic categorization model from unlabeled data. In other words, the parameters of the component densities are unknown and EM algorithm aims to estimate them from the patterns. The EM algorithm initially assumes random assignment of examples to categories. Then an initial probabilistic model is learned by estimating model parameters from this randomly labeled data. We then iterate over the following two steps until convergence:

- Expectation (E-step): Rescore every pattern given the current model, and probabilistically re-label the patterns based on these posterior probability estimates.
- Maximization (M-step): Re-estimate the model parameters from the probabilistically re-labeled data.

A practical description of this algorithm has been provided in Mitchell (1997).

2.2.1.5 Mode Seeking

One of the simplest ways to determine the modes in a dataset is to construct a histogram by portioning the pattern space into *k* non-overlapping regions. Regions with relatively high pattern frequency counts are the modes or cluster centers. In non-parametric density estimation, the clustering procedure searches for bins with large counts in a multidimensional histogram of the input patterns (Jain & Dubes, 1988). The regions of pattern space in which the patterns are the densest would represent the partition components. The regions that include fewer numbers of patterns separate the clusters.

As with other clustering methods, there are benefits and drawbacks. The advantage of the density estimation method is that it does not require knowing the number of clusters and their prior probabilities. The disadvantage of this approach is that the process of looking for the peaks and valleys in the histogram is difficult in more than a few dimensions and requires the user to identify the valleys in histograms for splitting interactively.

2.2.1.6 *k*-medoids

Instead of taking the mean value of the data points in a cluster, the *k*-medoids method represents a cluster with an actual data point that is the closest to the center of gravity of that cluster. Thus, the *k*-medoids method is less sensitive to noise and outliers than the *k*-means and the EM algorithms. This, however, requires a longer computational time. To determine which objects are good representatives of clusters, the *k*-medoids algorithm follows a cost function that dynamically evaluates any non-center data point against other existing data points.

2.2.1.6.1 Partitioning Around Medoids (PAM)

PAM (Kaufman & Rousseeuw, 1990) is one of the first *k*-medoids clustering algorithms which first selects the initial k cluster centers randomly within a data set of N objects. For every k cluster centers, PAM examines all non-center $(N - k)$ objects and tries to replace each of the centers with one of the $(N - k)$ objects that would reduce the square error the most. PAM works well when the number of data points is small. However, PAM is very costly, because for every $k \times (N - k)$ pairs PAM examines the (N

– k) data points to compute the cost function. Therefore, the total complexity is $O(k \times (N - k)^2)$.

2.2.1.6.2 CLARA

Because of the complexity inherent in PAM, an enhanced version of PAM was developed for use in large data sets. CLARA (Clustering LARge Applications) is a sampling-based method that was developed by Kaufman & Rousseeuw (1990). CLARA selects a small portion of data to represent all data points therein. It selects the medoids from these samples using PAM. The cost function is computed using the whole data set. The efficiency of CLARA depends on the sample size, S. CLARA searches for the best k-medoids among the sample of the data set. For more efficient results, CLARA draws multiple samples from the data set, runs PAM on each sample and returns the best clustering. The complexity of each iteration becomes $O(k \times S^2 + k \times (N - k))$ where S is the sample size, k is the number of clusters, and N is the number of data objects. It is important to note, when comparing this complexity with that of the standard PAM, the major distinction lies in the power of $N - k$, the largest value within the expression.

2.2.1.6.3 CLARANS

CLARANS (Clustering Large Applications based upon RANdomized Search) was proposed by Ng and Han (1994) to improve the quality and scalability of CLARA. This algorithm tries to find a better solution by randomly picking one of the k centers and replacing it with another randomly chosen object from the remaining objects. The goal of CLARANS is not to find the best set of data points that represent the cluster centers. Instead, it trades accuracy for efficiency and tries to find the local optimum. The randomized procedure applied by CLARANS is the strong point of this algorithm. Ng and Han (1994) have shown that CLARANS outperforms the PAM and CLARA algorithms, however, the computational complexity has been reported to be approximately $O(N^2)$ (Han et al., 2001). The clustering quality is highly dependent upon the sampling method employed. Ester et al. (1995) improved the performance of CLARANS using special clustering methods, such as R^*-trees.

2.2.1.7 Other partitional methods for large data sets

The main drawback of the CLARA and CLARANS algorithms is their requirement to hold the entire data set in the main memory. In general, large data mining applications do not allow the entire data set to be stored in the main memory, so clustering algorithms that can handle this situation are required. Some approaches were proposed to cluster the data from such sets that we explain two of them as follows.

1. *Divide and conquer approach*. The entire data set is stored in a secondary memory. The stored data is divided into p subsets. Each subset is clustered separately into k clusters. One or more representative samples from each of these clusters are stored individually. In the final step, all these p subset clusters are merged into k clusters to obtain a clustering of the entire set. This method can be extended to any number of levels (subsets); more subsets are necessary if the main memory size is small and the data set is very large (Murty and Krishna, 1980). The drawback of this algorithm is that it works well only when the points in each subset are realistically homogenous, e.g. in image data sets (Jain et al., 1999).

2. *Incremental clustering*. In this method the entire pattern matrix is stored in a secondary location from which data objects are loaded into main memory one at a time and assigned to existing clusters. Only the cluster representatives are stored in the main memory. Each new data is allocated to an existing cluster or assigned to a new cluster depending on criteria like the distance between the loaded data point and the cluster's representative. This method is naturally non-iterative, thus the time complexity requirement is as small as the memory requirement (Jain et al., 1999).

2.2.2 Hierarchical Methods

Hierarchical methods decompose the given set of data items forming a tree, which is called *dendrogram*. A dendrogram splits the dataset recursively into smaller subsets. A dendrogram can be formed in two ways:

1. The *Bottom-up* approach, also referred to as the *agglomerative* approach, starts with each object forming a distinct group. It successively merges the groups according to some measure, such as the distance between the centers of the groups, which continues until all of the groups are merged into one – the top most level of hierarchy.
2. The *Top-down* approach, also referred to as the *divisive* approach, starts with all the objects in the same cluster. In every successive iteration, a cluster is split into smaller groups according to some measure until each object is eventually in one cluster, or until a termination condition is met.
3. Hierarchical methods are popular in biological, social and behavioral systems, which often need to construct taxonomies. Due to rapidly increasing data

densities, dendrograms are impractical when the number of patterns exceeds a few hundred (Jain & Dubes, 1988). As a result, partitional techniques are more appropriate in the case of large data sets. The dendrogram can be broken at different levels to obtain different clusterings of the data (Jain et al., 1999).

2.2.2.1 Traditional Linkage Algorithms

The main steps of hierarchical agglomerative algorithms are as follows: We compute the proximity matrix including the distances between each pair of patterns. Each pattern is treated as a cluster in the first run. Using the proximity matrix we find the most similar pair of clusters and merge these two clusters into one. At this point, the proximity matrix is updated to imitate the merge operation. We continue this process until all patterns are in one cluster. Based on how the proximity matrix is updated a range of agglomerative algorithms can be designed (Jain et al., 1999).

1. *single-link* clustering: The similarity between one cluster and another cluster is equal to the greatest similarity between any member of one cluster and any member of another cluster. It is important to note that, by "greatest similarity," a smallest distance is implied.
2. *complete-link* clustering: The distance between one cluster and another cluster is equal to the greatest distance from any member of one cluster to any member of another cluster. Unlike the single-link algorithm, this is focused on the *lowest* similarity between clusters.
3. *average-link* clustering: The distance between one cluster and another cluster is equal to the average distance from any member of one cluster to any member of the another cluster.

2.2.2.2 BIRCH

The BIRCH (Balanced Iterative Reducing and Clustering) algorithm (Zhang et al., 1996) uses a hierarchical data structure, which is referred to as a CF-tree (Clustering-Feature-Tree) for incremental and dynamic clustering of data objects. The BIRICH algorithm represents data points as many small CF-trees and then performs clustering with these CF-trees as the objects. A CF is a triplet summarizing information about the sub-cluster in the CF-tree; CF = (*N, LS, SS*) where *N* denotes the number of objects in the sub-cluster, *LS* is the linear sum of squares of the data points, and *SS* is the sum of squares of the data points. Taken together, these three statistical measurements become the object for further pair-wise computation between any two sub-clusters (CF-trees). CF-trees are height-balanced trees that can be treated as sub-clusters. The BIRCH algorithm calls for two input factors to construct the CF-tree: the branching input factor *B* and threshold *T*. The branching parameter, *B,* determines the maximum number of child nodes for each CF node. The threshold, *T*, verifies the maximum diameter of the sub-cluster kept in the node (Han et al, 2001).

A CF tree is constructed as the data is scanned. Each point is inserted into a CF node that is most similar to it. If a node has more than *B* data points or its diameter exceeds the threshold *T*, BIRCH splits the CF nodes into two. After doing this split, if the parent node contains more than the branching factor *B,* then the parent node is rebuilt as well. The step of generating sub-clusters stored in the CF-trees can be viewed as a pre-clustering stage that reduces the total number of data to a size that fits in the main memory. The BIRCH algorithm performs a known clustering algorithm on the sub-cluster stored in the CF-tree. If *N* is the number of data points, then the computational complexity of the

BIRCH algorithm would be $O(N)$ because it only requires one scan of the data set – making it a computationally less expensive clustering method than hierarchical methods. Experiments have shown good clustering results for the BIRCH algorithm (Han et al, 2001). However, similar to many partitional algorithms it does not perform well when the clusters are not spherical in shape and also when the clusters have different sizes. This is due the fact that this algorithm employs the notion of diameter as a control parameter (Han et al, 2001). Clearly, one needs to consider both computational cost and geometrical constraints when selecting a clustering algorithm, even though real data sets are often difficult to visualize when first encountered.

2.2.2.3 CURE

The CURE (Clustering Using REpresentatives) algorithm (Guha et al., 1998) integrates different partitional and hierarchical clusters to construct an approach which can handle large data sets and overcome the problem of clusters with non-spherical shape and non-uniform size. The CURE algorithm is similar to the BIRCH algorithm and summarizes the data points into sub-clusters, then merges the sub-clusters that are most similar in a bottom-up (agglomerative) style. Instead of using one centroid to represent each cluster, the CURE algorithm selects a fixed number of well-scattered data points to represent each cluster (Han et al., 2001).

Once the representative points are selected, they are shrunk towards the gravity centers by a shrinking factor α which ranges between 0 and 1. This helps eliminate the effects of outliers, which are often far away from the centers and thus usually shrink more. After the shrinking step, this algorithm uses an agglomerative hierarchical method to perform the actual clustering. The distance between two clusters is the minimum distance between any representative points. Therefore, if $\alpha = 1$, then this algorithm will be a single link algorithm, and if $\alpha = 0$, then it would be equivalent to a centroid-based hierarchical algorithm (Guha & Rastogi, 1998). The algorithm can be summarized as follows:

1. Draw a random sample s from the data set.
2. Partition the sample, s, into p partitions (each of size $|s| / p$).
3. Using the hierarchical clustering method, cluster the objects in each sub-cluster (group) into $|s| / pq$ clusters, where q is a positive input parameter.
4. Eliminate outliers; if a cluster grows too slowly, then eliminate it.
5. Shrink multiple cluster representatives toward the gravity center by a fraction of the shrinking factor α.
6. Assign each point to its nearest cluster to find a final clustering.

This algorithm requires one scan of the entire data set. The complexity of the algorithm would be $O(N)$ where N is the number of data points. However, the clustering result depends on the input parameters $|s|$, p, and α. Tuning these parameters can be difficult and requires some expertise, making this algorithm difficult to recommend (Han et al. 2001).

2.3 Feature Selection and Extraction

Feature extraction and selection is a very important task in the classification or clustering process. *Feature selection* is the procedure of discovering the most effective subset of the original features to use in classification/clustering. *Feature extraction* is the process of transforming the input features to produce new relevant features. Either or both of these techniques can be used to obtain an appropriate set of features to use in classification or clustering. Why do we need to use feature selection or extraction? We can denote the benefits (Jain, 2000) of feature selection and extraction as follows:

2.3.1 Minimizing the cost

In many real world applications, feature measurement is very costly, especially with a large sample size. Pei, et al. (1998) presented in the context of a biological pattern classification that the most important 8 out of 96 features gives 90% classification accuracy. They showed that feature selection has a great potential effect in minimizing the cost of extracting features and maintaining good classification results.

2.3.2 Data Visualization

For explanatory purposes it is useful to project high dimensional data down to two or three dimensions. The main concern is protecting the distance information and deployment of the original data in two or three dimensions. The traditional approach in data visualization is linear projection. A more convenient choice is projecting the data onto a graph (Mori 1998). "*Chernoff Faces*" project the feature space onto cartoon faces, by which one can visualize more than three features (Chernoff, 1973).

2.3.3 Dimensionality Reduction

Based on an ideal situation where we have an infinite number of training samples, classification would be more accurate when we have more features because generally, more features gives more information. Nevertheless, in real applications with finite sample sizes, the maximum performance is inversely proportional to the number of features (Duda & Heart, 1973).

The demand for a large number of samples grows exponentially with the dimensionality of the feature space. This is due to the fact that as the dimensionality grows, the data objects becomes increasingly sparse in the space it occupies. Therefore, for classification, this means that there are not enough sample data to allow for reliable assignment of a class to all possible values; and for clustering, the definition of density and distance among data objects, which is critical in clustering, becomes less meaningful (Duda & Heart, 1973).

This limitation is referred to as the "*curse of dimensionality*" (Duda et al., 2001). Trunk (1979) has represented the curse of dimensionality problem through an exciting and simple example. He considered a 2-class classification problem with equal prior probabilities, and a *d*-dimensional multivariate Gaussian distribution with the identity covariance matrix for each class. Trunk showed that the probability of error approaches the maximum possible value of 0.5 for this 2-class problem. This study demonstrates that one cannot increase the number of features when the parameters for the class-conditional density are estimated from a finite number of training samples. Therefore, when the training sets are limited, one should try to select a small number of salient features.

This puts a limitation on non-parametric decision rules such as *k*-nearest neighbor. Therefore it is often desirable to reduce the dimensionality of the space by finding a new set of bases for the feature space.

2.3.4 Feature Selection

A thorough review of feature selection has been presented in Jain (2000). Jain et al. (1997) presented a taxonomy of available feature selection algorithms based on pattern recognition or ANN, sub-optimal or optimal, single solution or multi-solution, and deterministic or stochastic.

What are the criteria for choosing a feature subset? A subset of features might be chosen in regards to the following points:

1. The *relevance* to the classification result: that is, we remove the irrelevant features based on prior knowledge of the classification task. Langley (1994) has a useful review on relevance and feature selection.
2. The *correlation* with the other features: High correlation among features will add no more efficiency to classification (Harrell & Frank, 2001). That is, if two features are highly correlated, one of the features is redundant, even though it is relevant.

John et al. (1994) has presented the definitions of *Strong Relevance* and *Weak Relevance* by considering the correlations among feature samples. Hall and Smith (1998) formulated a measure of "*Goodness of feature*" as follows:

> *"Good feature subsets contain features highly correlated (predictive of) with the class, yet uncorrelated with (not predictive of) each other."*

2.3.5 Feature Extraction

Feature extraction can be either a linear or non-linear transformation of the original feature space. Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are the most commonly used techniques for feature extraction. PCA is an unsupervised technique which is intended for feature extraction, while LDA is supervised.

The idea of the PCA is to preserve the maximum variance after transformation of the original features into new features. The new features are also referred to as “principal components” or “factors.” Some factors carry more variance than other, but if we limit the total variance preserved after such a transformation to some portion of the original variance, we can generally keep a smaller number of features. PCA performs this reduction of dimensionality by determining the covariance matrix. After the PCA transformation in a d-dimensional feature space the m ($m < d$) largest eigenvalues of the $d \times d$ covariance matrix are preserved. That is, the uncorrelated m projections in the original feature space with the largest variances are selected as the new features, thus the dimensionality reduces from d to m (Duda al et., 2001).

LDA uses the same idea but in a supervised learning environment. That is, it selects the m projections using the criterion that maximizes the inter-class variance while minimizing the intra-class variance (Duda al et., 2001). Due to supervision, LDA is more efficient than PCA for feature extraction.

As explained in PCA, m uncorrelated linear projections are selected as the extracted features. Nevertheless, from the statistical point of view, for two random variables that do not hold normal distribution, uncorrelated between each other does not necessarily lead to independent between each other. Therefore, a novel feature extraction technique, Independent Component Analysis (ICA) has been proposed to handle the non-Gaussian distribution data sets (Comon, 1994). Karhunen (1997) gave an simple example when the axes found by ICA is different than those found by PCA for two features uniformly distributed inside a parallelogram.

2.4 Summary

A body of literature was briefly explained which deals with the different problems involved in data mining for performing classification and clustering upon a web-based educational data. The major clustering and classification methods are briefly explained, along with the concepts, benefits, and methods for feature selection and extraction. In the next chapter, we design and implement a series of pattern classifiers in order to compare their performance for a data set from the LON-CAPA system. This experiment provides an opportunity to study how classification methods could be put into practice for future web-based educational systems.

Chapter 3 Data Representation and Assessment Tools in LON-CAPA

This chapter provides information about the structure of LON-CAPA data namely: its data retrieval process, how we provide assessment tools in LON-CAPA on many aspects of teaching and learning process. Our ability to detect, to understand, and to address student difficulties is highly dependent on the capabilities of the tool. Feedback from numerous sources has considerably improved the educational materials, which is a continuing task.

3.1 Data Acquisition and Extracting the Features

3.1.1 Preprocessing student database

Preprocessing and finding the useful student data and segmenting may be a difficult task. As mentioned earlier, LON-CAPA has two kinds of large data sets: 1) Educational resources such as web pages, demonstrations, simulations, and individualized problems designed for use on homework assignments, quizzes, and examinations; 2) Information about users, who create, modify, assess, or use these resources.

The original data are stored with escape sequence codes as shown in Figure 3.1:

```
1007070627:msul1:1007070573%3a%2fres%2fadm%2fpages%2fgrds%2egif%3aminaeibi%3amsu%26100707
0573%3a%2fres%2fadm%2fpages%2fstat%2egif%3aminaeibi%3amsu%261007070574%3amsu%2fmmp%2flabq
uiz%2flabquiz%2esequence___1___msu%2fmmp%2flabquiz%2fnewclass%2ehtml%3aminaeibi%3amsu%261
007070589%3amsu%2fmmp%2flabquiz%2flabquiz%2esequence___5___msu%2fmmp%2flabquiz%2fproblems
%2fquiz2part2%2eproblem%3aminaeibi%3amsu%261007070606%3a%2fadm%2fflip%3aminaeibi%3amsu%26
1007070620%3a%2fadm%2fflip%3aminaeibi%3amsu%261007070627%3a%2fres%2fadm%2fpages%2fs%2egif
%3aminaeibi%3amsu%261007070627%3a%2fadm%2flogout%3aminaeibi%3amsu
```

Figure 3.1 A sample of stored data in escape sequence code

To sense the data we use the following Perl script function as shown in Figure 3.2

```
 my $str; my $line;
open (LOG ,$file);
while ($line =<LOG>) {
   my ($dumptime,$host,$entry)=split(/\:/,$line);
   my $str = unescape($entry);
   my ($time,$url,$usr,$domain,$store,$dummy)=split(/\:/,$str);
   my $string = escape($store);
   foreach(split(/\&/,$string)){
       print "$time $url $usr domain \n";
   }
}

sub unescape {
       my $str=shift;
       $str =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",hex($1) )/eg;
       return $str;
}
```

Figure 3.2 Perl scrip code to retrieve stored data

After passing the data from this filter we have the following results as shown in Figure 3.3:

```
1007070573 /res/adm/pages/grds.gif minaeibi /res/adm/pages/stat.gif
1007670091 /res/adm/pages/grds.gif minaeibi /adm/flip
1007676278
msu/mmp/labquiz/labquiz.sequence___2___msu/mmp/labquiz/problems/quiz1part1.problem
1007743917 /adm/logout minaeibi
1008203043 msu/mmp/labquiz/labquiz.sequence___1___msu/mmp/labquiz/newclass.html minaeibi
1008202939 /adm/evaluate minaeibi /adm/evaluate
1008203046 /res/adm/pages/g.gif minaeibi /adm/evaluate
1008202926 /adm/evaluate minaeibi
```

Figure 3.3 A sample of retrieved data from activity log

The student data restored from .db files from a student directory and fetched into a hash table. The special hash keys "*keys*", "*version*" and "*timestamp*" were obtained from the hash. The *version* will be equal to the total number of versions of the data that have been stored. The *timestamp* attribute is the UNIX time the data was stored. *keys* is available in every historical section to list which keys were added or changed at a specific historical revision of a hash. We extract some of the features from a structured homework data, which is stored as particular URL’s. The structure is shown in figure 3.4.

```
resource.partid.opendate      #unix time of when the local machine should let the
                      #student in
resource.partid.duedate       #unix time of when the local machine should stop
                      #accepting answers
resource.partid.answerdate    #unix time of when the local machine should
                              #provide the correct answer to the student
resource.partid.weight     # points the problem is worth
resource.partid.maxtries   # maximum number of attempts the student can have
resource.partid.tol # lots of possibilities here
                    # percentage, range (inclusive and exclusive),
                 # variable name, etc
                    # 3%
                    # 0.5
                    # .05+
                    # 3%+
                    # 0.5+,.005
resource.partid.sig  # one or two comma sepearted integers, specifying the
                     # number of significatn figures a student must use
resource.partid.feedback # at least a single bit (yes/no) may go with a
                         # bitmask in the future, controls whether or not
                         # a problem should say "correct" or not
resource.partid.solved # if not set, problem yet to be viewed
                # incorrect_attempted == incorrect and attempted
                # correct_by_student == correct by student work
                # correct_by_override == correct, instructor override
                # incorrect_by_override == incorrect, instructor override
                # excused == excused, problem no longer counts for student
                # '' (empty) == not attempted
                # ungraded_attempted == an ungraded answer has been
                                          sumbitted and stored
resource.partid.tries  # positive integer of number of unsuccessful attempts
                # made, malformed answers don't count if feedback is
                # on
resource.partid.awarded       # float between 0 and 1, percentage of
                      # resource.weight that the stundent earned.
resource.partid.responseid.submissons
                      # the student submitted string for the part.response
resource.partid.responseid.awarddetail
                     # list of all of the results of grading the submissions
                     # in detailed form of the specific failure
                 # Possible values:
                     # EXACT_ANS, APPROX_ANS : student is correct
                     # NO_RESPONSE : student submitted no response
                     # MISSING_ANSWER : student submitted some but not
                     #                   all parts of a response
                     # WANTED_NUMERIC : expected a numeric answer and
                     #                   didn't get one
                 # SIG_FAIL : incorrect number of Significant Figures
                     # UNIT_FAIL : incorrect unit
                     # UNIT_NOTNEEDED : Submitted a unit when one shouldn't
                     # NO_UNIT : needed a unit but none was submitted
                 # BAD_FORMULA : syntax error in submitted formula
                     # INCORRECT : answer was wrong
                     # SUBMITTED : submission wasn't graded
```

Figure 3.4 Structure of stored data in activity log and student data base

For example, the result of solving homework problem by students could be extracted from *resource.partid.solved*, the total number of the students for solving the problem could be extracted from `resource.partid.tries`, and so forth. One of the difficult phases to data mining in the LON-CAPA system is gathering the student and course data, which are distributed in several locations. Finding the relevant data and segmentation phase is complicated as well.

3.1.2 Preprocessing Activity Log

LON-CAPA records and dynamically organizes a vast amount of information on students' interaction with and understanding of these materials. Since LON-CAPA logs every activity of every student who has used online educational resources and their recorded paths, the activity.log usually grows faster when students have more access to the educational resources. A sample of different types of data, which are logged in activity.log after a preprocessing phase, is shown in figure 3.5.

```
144) 1010955846: studentX --> /adm/navmaps eb. This information is stored in an
“activity.log” which is located in a course’s directory. The data stored in the
activity.log includes user name, time and resource URL.
145) 1010955205: studentX --> /res/msu/mmp/kap14/picts/beta_eqn.gif
147) 1010955988: studentX --> /adm/navmaps
148) 1010955998: studentX --> msu/mmp/kap14/kap14.sequence___5___msu/mmp/kap14/cd396.htm
149) 1010955999: studentX --> /res/msu/mmp/kap14/picts/velocity_eqn3.gif
150) 1010956000: studentX --> /res/msu/mmp/kap14/picts/time_eqn.gif
151) 1010954609: studentX --> /res/adm/pages/grds.gif
152) 1010954611: studentX --> /res/msu/mmp/wordproc.gif
153) 1010954626: studentX --> /res/adm/pages/i.gif
154) 1010955717: studentX --> msu/mmp/kap14/kap14.sequence___1___msu/mmp/kap14/cd392.htm
155) 1010955717: studentX --> /res/msu/mmp/kap14/picts/backsoun.gif
156) 1010955920: studentX --> msu/mmp/kap14/kap14.sequence___3___msu/mmp/kap14/cd394.htm
157) 1010955921: studentX --> /res/msu/mmp/gifs/demo.gif
163) 1010955754: studentX --> msu/mmp/kap14/kap14.sequence___2___msu/mmp/kap14/cd393.htm
164) 1010955756: studentX --> /res/msu/mmp/kap14/picts/asound.jpg
166) 1010955999: studentX --> /res/msu/mmp/gifs2/example.gif
173) 1010955687: studentX --> /res/adm/pages/u.gif
174) 1010955688: studentX --> /res/adm/pages/e.gif
175) 1010956528: studentX -->
msu/mmp/kap14/kap14.sequence___33___msu/mmp/kap14/problems/cd418a.problem
176) 1010956536: studentX -->
msu/mmp/kap14/kap14.sequence___33___msu/mmp/kap14/problems/cd418a.problem
178) 1010956536: studentX -->
msu/mmp/kap14/kap14.sequence___33___msu/mmp/kap14/problems/cd418a.problem
```

Figure 3.5 A sample of extracted Activity.log data

3.1.3 Extractable Features

An essential step to perform classification is selecting the features used for classification. The following features are examples of those stored by the LON-CAPA system:

- Total number of correct answers.
- Getting the problem right on the first try.
- Number of attempts before correct answer is derived.
- Total time that passed from the first attempt, until the correct solution was demonstrated, regardless of the time spent logged in to the system. Also, the time at which the student got the problem correct relative to the due date.
- Total time spent on the problem regardless of whether they got the correct answer or not. Total time that passed from the first attempt through subsequent attempts until the last submission was demonstrated.
- Participating in the communication mechanisms, versus those working alone. LON-CAPA provides online interaction both with other students and with the instructor.
- Reading the supporting material before attempting homework vs. attempting the homework first and then reading up on it.
- Submitting a lot of attempts in a short amount of time without looking up material in between, versus those giving it one try, reading explanatory/supportive material, submitting another one, and so forth.
- Giving up on a problem versus students who continued trying up to the deadline.

- Time of the first log on (beginning of assignment, middle of the week, last minute) correlated with the number of submissions or number of solved problems.

These features enable LON-CAPA to provide many assessments tools for instructors as it will be explained in the next section.

3.2 Feedback to the instructor from online homework

LON-CAPA has enabled instructors to efficiently create and distribute a wide variety of educational materials, assignments, assessments, etc. These include numerous types of formative conceptual and algorithmic exercises for which prompt feedback and assistance can be provided to students as they work on assigned tasks. This section presents recent developments that allow rapid interpretation of such data in identifying students' misconceptions and other areas of difficulty, so that concurrent or timely corrective action can be taken. This information also facilitates detailed studies of the educational resources used and can lead to redesign of both the materials and the course.

3.2.1 Feedback tools

While several meta-analyses of the effects of assessment with immediate feedback to the student on their learning are positive (Mason & Bruning, 2003; Azevedo & Bernard 1995), the range of effect size is considerable (Bonham et al., 2001), and can even be negative (Mason & Bruning, 2003; Bransford et al., 2000; Kluger & DeNisi, 1996; Kluger & DeNisi, 1998). Even within our own model systems CAPA, LectureOnline, and LON-CAPA, when used just for homework, a range of partly contradictory observations were made (Kotas, 2000; Pascarella, 2004). The timely feedback is crucial for ensuring effective use.

As it has been mentioned earlier LON-CAPA do record all information transmitted to and from the student. That large amount of data, especially in large courses, is much too voluminous for the faculty to interpret and use without considerable pre-processing. We discuss functions that make that vast amount of data useful in a timely fashion. The instructor can then give students useful feedback, either promptly enough that student can benefit while still working on current task, or at a later date to clarify misconceptions and address lack of understanding. A preliminary report on some of this work was presented in Albertelli et al., (2002).

LON-CAPA outperforms many web-based education systems in three important aspects relevant to the current discussion.

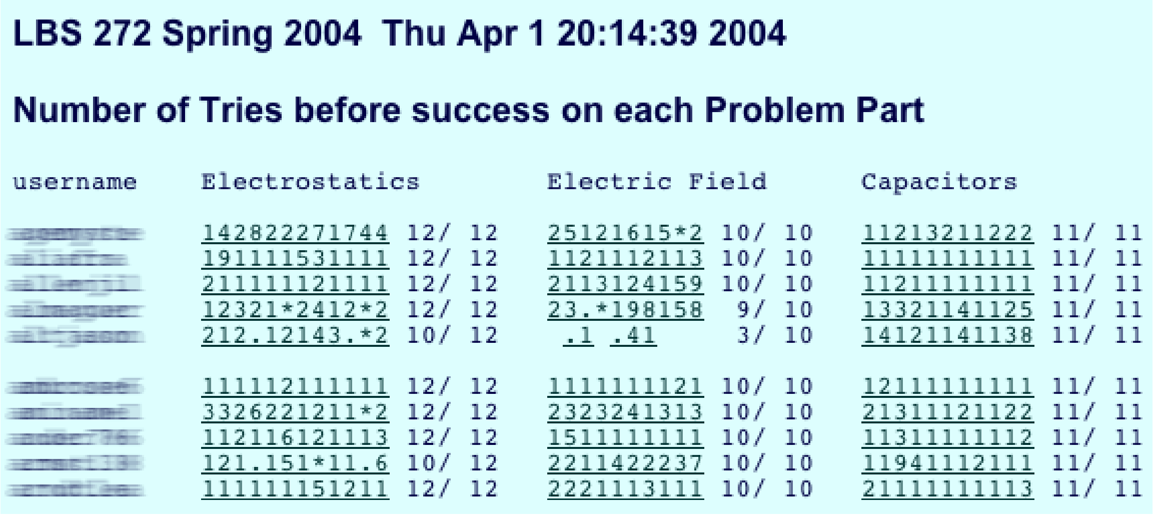
1. The first is its ability to individualize problems, both algorithmic numerical exercises as well as problems that are qualitative and conceptual so that numbers, options, images, etc. differ from student to student. (Kashy et al., 1995).
2. The second is in the tools provided that allow instructor to collaborate in the creation and sharing of content in a fast and efficient manner, both within and across institutions, thus implementing the initial goals of the WWW[3].
3. And the third is its one-source multiple target capabilities: that is, its ability to automatically transform one educational resource, for example a numerical or conceptual homework question, into a format suitable for multiple uses.

[3] See http://www.w3.org/History/19921103-hypertext/hypertext/WWW/Proposal.html and also http://www.w3.org/History/19921103-hypertext/hypertext/WWW/DesignIssues/Multiuser.html

3.2.2 Student Evaluation

An important task of the feedback tools for the instructor is to help identify the source of difficulties and the misconceptions students have about a topic. There are basically three ways to look at such homework data: by student, by problem, or cross-cutting (per student, per problem).

The amount of data gathered from large enrollment courses (200-400 students) with over 200 randomizing homework problems, each of them allowing multiple attempts, can be overwhelming. Figure 3.6 shows just a small excerpt of the homework performance in an introductory physics course, students in the rows, problems in the columns, each character representing one online homework problem for one student. A number shown is the number of attempts it took that particular student to get that particular problem correct ("*" means more than nine attempts), "." denotes an unsolved problem, blank an un-attempted problem. This view is particularly useful ahead of the problem deadline, where columns with a large number of dots or blank spaces indicate problems that the students have difficulties with.



LBS 272 Spring 2004 Thu Apr 1 20:14:39 2004

Number of Tries before success on each Problem Part

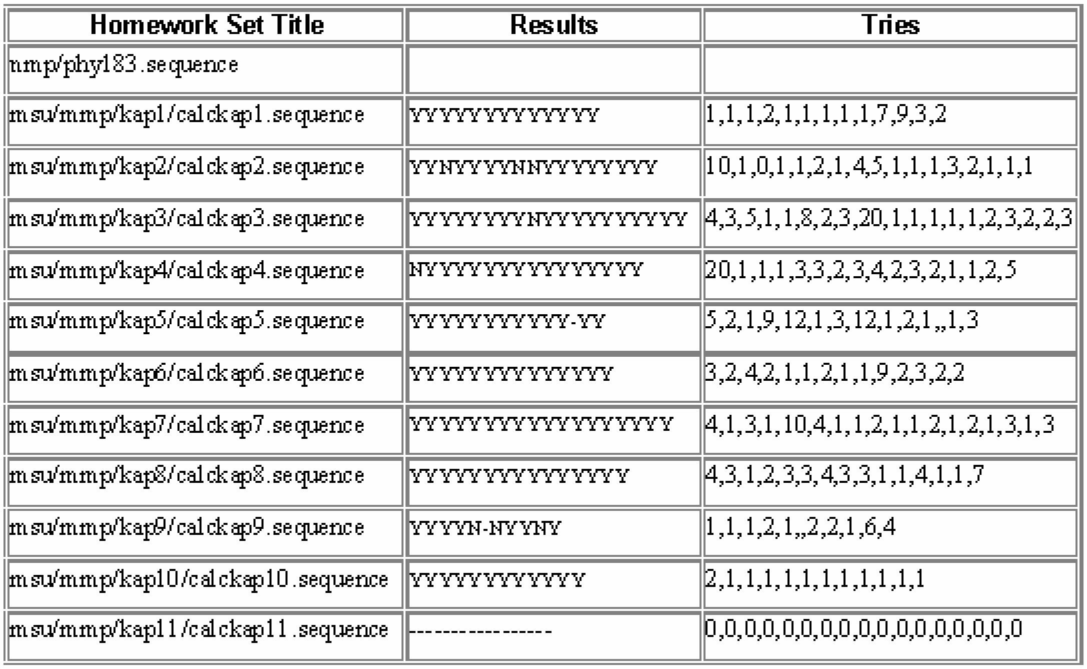
```
username     Electrostatics         Electric Field       Capacitors

[redacted]   142822271744 12/ 12    25121615*2 10/ 10    11213211222 11/ 11
[redacted]   191111531111 12/ 12    1121112113 10/ 10    11111111111 11/ 11
[redacted]   211111121111 12/ 12    2113124159 10/ 10    11211111111 11/ 11
[redacted]   12321*2412*2 12/ 12    23.*198158  9/ 10    13321141125 11/ 11
[redacted]   212.12143.*2 10/ 12     .1  .41    3/ 10    14121141138 11/ 11

[redacted]   111112111111 12/ 12    1111111121 10/ 10    12111111111 11/ 11
[redacted]   3326221211*2 12/ 12    2323241313 10/ 10    21311121122 11/ 11
[redacted]   112116121113 12/ 12    1511111111 10/ 10    11311111112 11/ 11
[redacted]   121.151*11.6 10/ 12    2211422237 10/ 10    11941112111 11/ 11
[redacted]   111111151211 12/ 12    2221113111 10/ 10    21111111113 11/ 11
```

1..9: correct by student in 1..9 submissions	*: correct by student in more than 9 submissions
+: correct by override	-: incorrect by override
.: incorrect attempted	#: ungraded attempted
‘ ‘: not attempted	x: excused

Figure 3.6 A small excerpt of the performance overview for a small introductory physics class

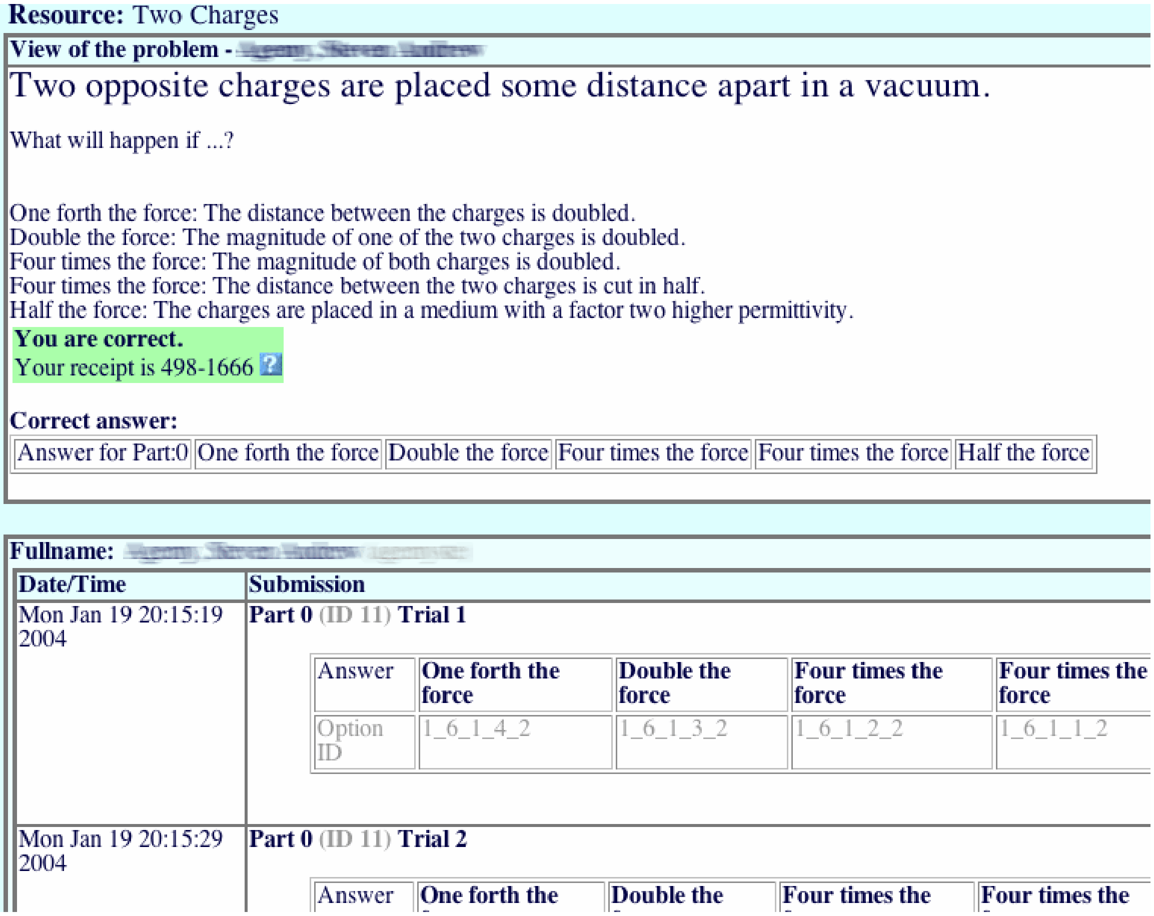
We extract from student data some reports of the current educational situation of every student as shown in table 3.1. A ‘Y’ shows that the student has solved the problem and an ‘N’ shows a failure. A ‘-‘ denotes an un-attempted problem. The numbers in the right column show the total number of submissions of the student in solving the corresponding problems.

Table 3.1 A sample of a student homework results and submissions



Homework Set Title	Results	Tries
mmp/phy183.sequence		
msu/mmp/kap1/calckap1.sequence	YYYYYYYYYYYYY	1,1,1,2,1,1,1,1,1,7,9,3,2
msu/mmp/kap2/calckap2.sequence	YYNYYYYNNYYYYYYYY	10,1,0,1,1,2,1,4,5,1,1,1,3,2,1,1,1
msu/mmp/kap3/calckap3.sequence	YYYYYYYYNYYYYYYYYYY	4,3,5,1,1,8,2,3,20,1,1,1,1,1,2,3,2,2,3
msu/mmp/kap4/calckap4.sequence	NYYYYYYYYYYYYYYY	20,1,1,1,3,3,2,3,4,2,3,2,1,1,2,5
msu/mmp/kap5/calckap5.sequence	YYYYYYYYYYY-YY	5,2,1,9,12,1,3,12,1,2,1,,1,3
msu/mmp/kap6/calckap6.sequence	YYYYYYYYYYYYYY	3,2,4,2,1,1,2,1,1,9,2,3,2,2
msu/mmp/kap7/calckap7.sequence	YYYYYYYYYYYYYYYYYY	4,1,3,1,10,4,1,1,2,1,1,2,1,2,1,3,1,3
msu/mmp/kap8/calckap8.sequence	YYYYYYYYYYYYYYY	4,3,1,2,3,3,4,3,3,1,1,4,1,1,7
msu/mmp/kap9/calckap9.sequence	YYYYN-NYYNY	1,1,1,2,1,,2,2,1,6,4
msu/mmp/kap10/calckap10.sequence	YYYYYYYYYYYY	2,1,1,1,1,1,1,1,1,1,1,1
msu/mmp/kap11/calckap11.sequence	----------------	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

For a per-student view, each of the items in the table in Figure 3.6 is clickable and shows both the students' version of the problem (since each is different), and their previous attempts. Figure 3.7 is an example of this view, and indicates that in the presence of a medium between the charges, the student was convinced that the force would increase, but also that this statement was the one he was most unsure about: His first answer was that the force would double; no additional feedback except "incorrect" was provided by the system. In his next attempt, he would change his answer on only this one statement (indicating that he was convinced of his other answers) to "four times the force" – however, only ten seconds passed between the attempts, showing that he was merely guessing by which factor the force increased.



Resource: Two Charges

View of the problem - [redacted]

Two opposite charges are placed some distance apart in a vacuum.

What will happen if ...?

One forth the force: The distance between the charges is doubled.
Double the force: The magnitude of one of the two charges is doubled.
Four times the force: The magnitude of both charges is doubled.
Four times the force: The distance between the two charges is cut in half.
Half the force: The charges are placed in a medium with a factor two higher permittivity.

You are correct.
Your receipt is 498-1666 ?

Correct answer:

Answer for Part:0	One forth the force	Double the force	Four times the force	Four times the force	Half the force

Fullname: [redacted]

Date/Time	**Submission**
Mon Jan 19 20:15:19 2004	**Part 0** (ID 11) **Trial 1**
Mon Jan 19 20:15:29 2004	**Part 0** (ID 11) **Trial 2**

Trial 1:

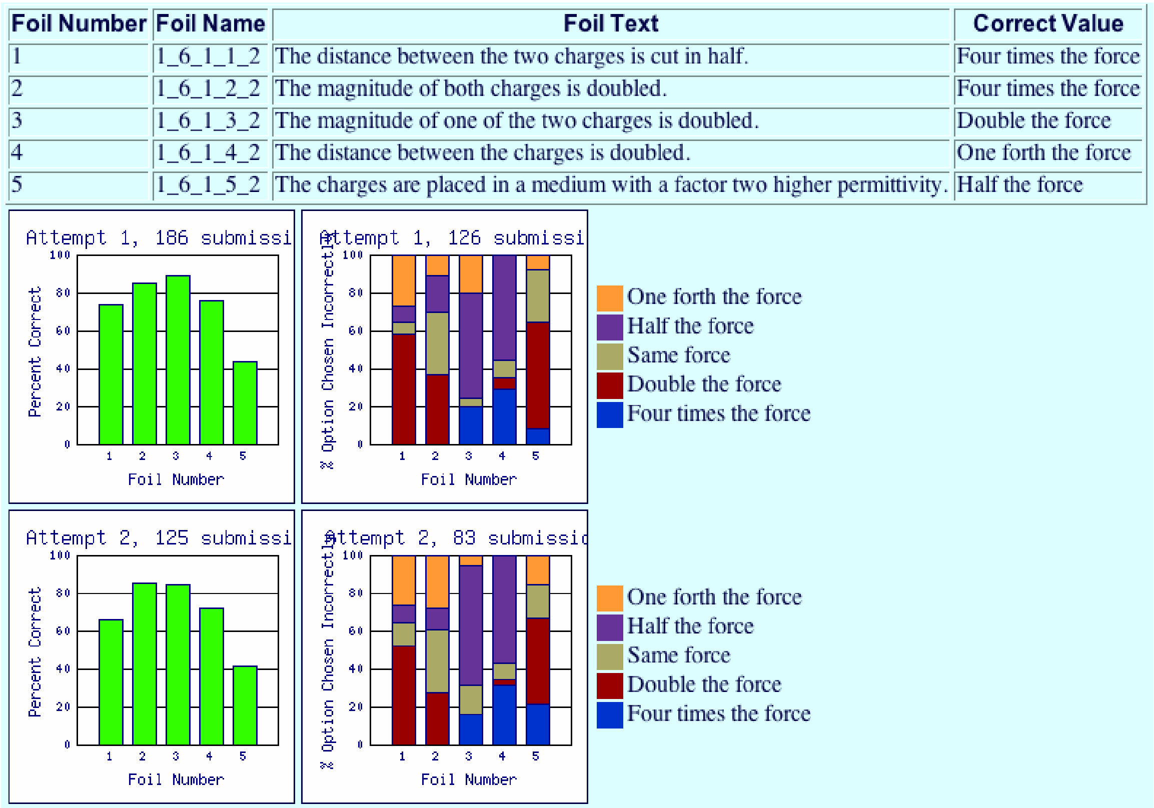
Answer	**One forth the force**	**Double the force**	**Four times the force**	**Four times the force**
Option ID	1_6_1_4_2	1_6_1_3_2	1_6_1_2_2	1_6_1_1_2

Trial 2:

Answer	**One forth the**	**Double the**	**Four times the**	**Four times the**

Figure 3.7 Single-student view of a problem

The per-problem view Figure 3.8 shows which statements were answered correctly course-wide on the first and on the second attempt, respectively, the graphs on the right which other options the students chose if the statement was answered incorrectly. Clearly, students have the most difficulty with the concept of how a medium acts between charges, with the absolute majority believing the force would increase, and about 20% of the students believing that the medium has no influence – this should be dealt with again in class.



Foil Number	Foil Name	Foil Text	Correct Value
1	1_6_1_1_2	The distance between the two charges is cut in half.	Four times the force
2	1_6_1_2_2	The magnitude of both charges is doubled.	Four times the force
3	1_6_1_3_2	The magnitude of one of the two charges is doubled.	Double the force
4	1_6_1_4_2	The distance between the charges is doubled.	One forth the force
5	1_6_1_5_2	The charges are placed in a medium with a factor two higher permittivity.	Half the force

Figure 3.8 Compiled student responses to a problem

The simplest function of the statistics tools in the system is to quickly identify areas of student difficulties. This is done by looking at the number of submissions students require in reaching a correct answer, and is especially useful early after an assignment is given. A high degree of failure indicates the need for more discussion of the topic before the due date, especially since early responders are often the more dedicated and capable students in a course. Sometimes a high degree of failure has been the result of some ambiguity in wording or, mostly in newly authored problem resources, the result of errors in their code. Difficulty is then ‘sky high’. Quick detection allows correction of the resource, often before most students have begun the assignment. Figure 3.9 shows a plot of the ratio of number of submissions to number of correct responses for 17 problems, from a weekly assignment before it was due. About 15% of the 400 students in an introductory physics course had submitted part or most of their assignment.

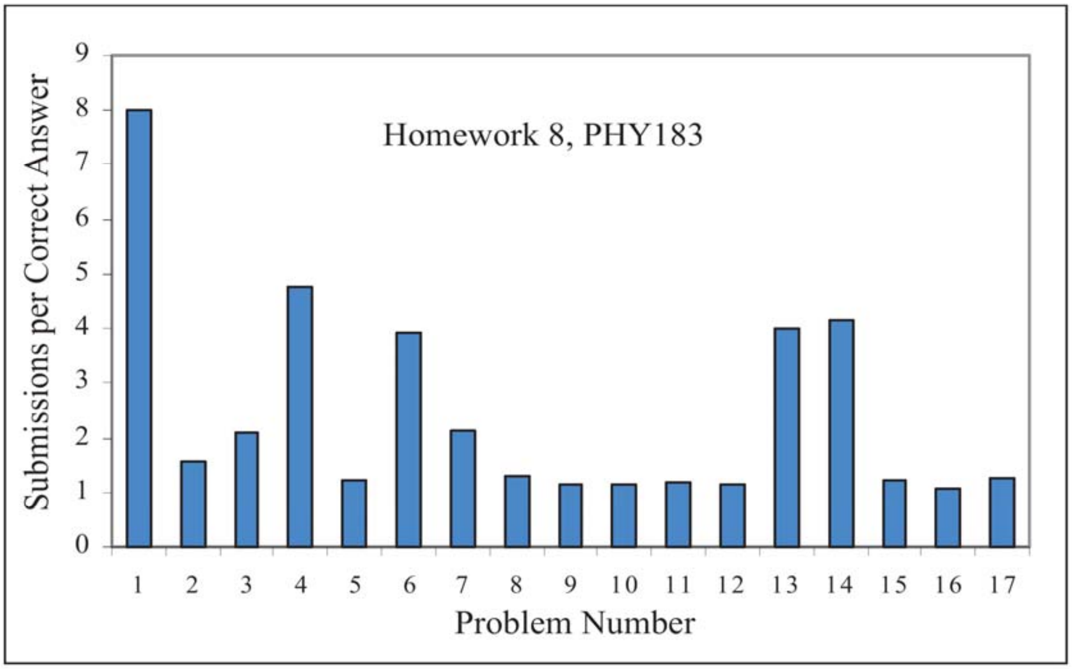


Figure 3.9 One Early Measure of a Degree of Difficulty

The data of Figure 3.9 is also available as a table which also lists the number of students who have submissions on each problem. Figure 3.9 shows that five of the questions are rather challenging, each requiring more than 4 submissions per success on average. Problem 1 requires a double integral in polar coordinates to calculate a center of mass. Problem 14 is a qualitative conceptual question with six parts and where it is more likely that one part or another will be missed. Note that incorrect use of a calculator or incorrect order of operation in a formula would not be detected in Figure 3.9 because of their relatively low occurrence. Note also that an error in the unit of the answer or in the formatting of an answer is not counted as a submission. In those instances, students re-enter their data with proper format and units, an important skill that students soon acquire without penalty.

3.2.3 Conceptual Problems

An important task of the feedback tools for the instructor is to help identify the source of difficulty in numerical algorithmic questions, but it also allows for the identification of misconceptions students may have on qualitative questions. Student responses to two qualitative exercises, one from physics and the second from vector math, illustrate the way that the analysis tool detects difficulties and their source, specific misconceptions. The physics question is Problem 14 from assignment 8 (Figure 3.12), which as indicated above, had five days before it was due. As shown in Figure 3.9 that problem averaged at that time slightly more than 4 submissions per successful solution. There were 50 correct solutions as a result of 208 submissions by 74 students. The order in which the six statements are presented varies among students. Each statement is selected randomly from one of the six concept groups. Each concept group focuses on a particular aspect in the question. Success rate on each concept for the initial submission is shown in Figure 3.10.

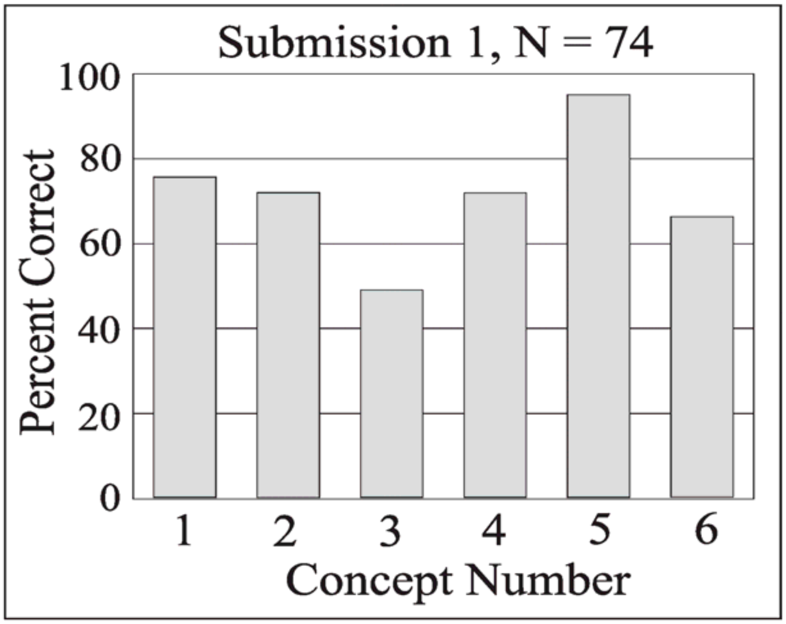


Figure 3.10 Success (%) in Initial Submission for Selecting the Correct Answer to Each of Six 'Concept' Statements

While concept '3' is quite clearly the most misunderstood, there is also a large error rate for concepts '2', 4' and '6'. About one third of the students succeeded on their first submission for all six concepts groups and thus earned credit on their first submission. This can be seen by looking at the decreasing number of submissions from Figure 3.10 to Figure 3.11. Note the pattern in the initial submissions persists in subsequent submissions with only minor changes.

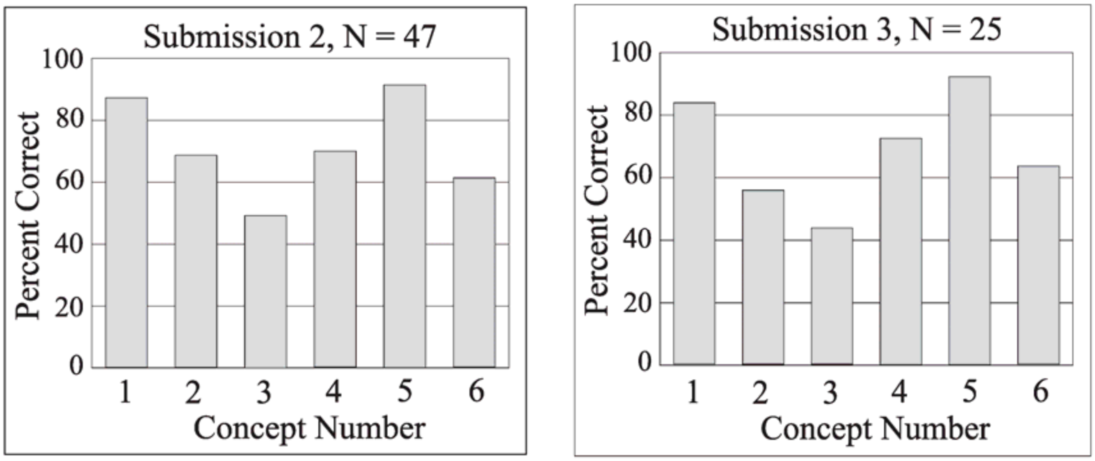
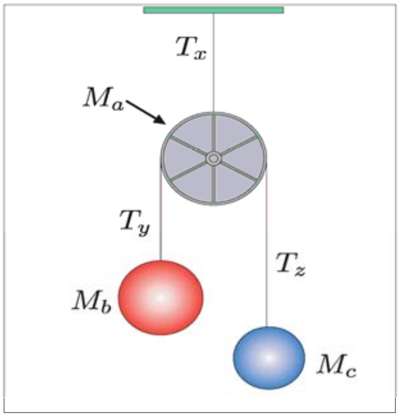


Figure 3.11 Success Rate on Second and Third Submissions for Answers to Each of Six 'Concept' Statements

The text of the problem corresponding to the data in Figures 3.10 and 3.11 is shown in Figure 3.12.



A frictionless pulley with mass M_a is attached to the ceiling, in a gravity field g. Mass M_c is smaller than mass M_b. The tensions T_x, T_y, T_z, and the constant g are magnitudes. For each statement, select a response.

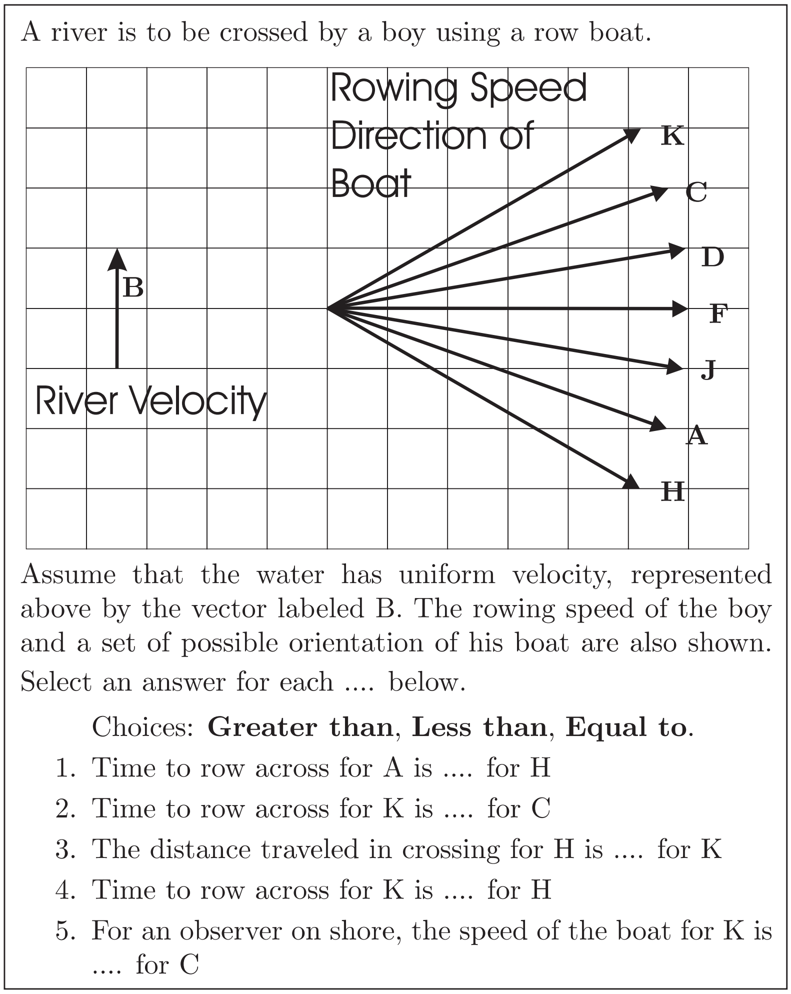
Choices: **Greater than**, **Less than**, **Equal to**, **True**, **False**.

1. T_z is .... T_y.
2. $(M_b+M_c+M_a)$g is .... T_x.
3. T_y is .... (M_b)g.
4. The magnitude of the acceleration of M_b is .... that of M_c.
5. The center-of-mass of M_b, M_c, and M_a accelerates.
6. T_x is .... T_z+T_y.

Figure 3.12 Randomly Labeled Conceptual Physics Problem

The labels in the problem are randomly permuted. In the version of the problem shown in Figure 3.12 the first question is to compare tension T_z to T_y. It is the most commonly missed statement, corresponding to concept ‘3’ of Figures 3.10 and 3.11. The incorrect answer given by over 90% of the students is that the two tensions are equal, which would be the answer for a pulley with negligible mass. That had been the case in an assignment two weeks earlier. This error was addressed by discussion in lecture and by a demonstration showing the motion for a massive pulley with unequal masses. This quickly impacted the subsequent response pattern. Note that solutions to the versions of the problems use as illustrations are given at the end of this section. (Solution to Figure 3.12: 1-less, 2-greater, 3-less, 4-equal, 5-true, 6-greater)

The next example is shown in Figure 3.13. It deals with the addition of two vectors. The vectors represent the possible orientations and rowing speed of a boat and the velocity of water. Here also the labeling is randomized so both the image and the text vary for different students. Students are encouraged to discuss and collaborate, but cannot simply copy from each other (Solution to Figure 3.13: 1-less, 2-greater, 3-less, 4-equal, 5-greater).



A river is to be crossed by a boy using a row boat.

Assume that the water has uniform velocity, represented above by the vector labeled B. The rowing speed of the boy and a set of possible orientation of his boat are also shown. Select an answer for each .... below.

Choices: **Greater than**, **Less than**, **Equal to**.

1. Time to row across for A is .... for H
2. Time to row across for K is .... for C
3. The distance traveled in crossing for H is .... for K
4. Time to row across for K is .... for H
5. For an observer on shore, the speed of the boat for K is .... for C

Figure 3.13 Vector Addition Concept Problem

The upper graphic of Figure 3.14 shows once again the success rate of 350 students on their initial submission, but this time in more detail showing all the possible statements. There are two variations for the first three concepts and four for the last two.

The lower graph in Figure 3.14 illustrates the distribution of incorrect choices for the 282 students who did not get earn credit for the problem on their first submission. The stacked bars show the way each statement was answered incorrectly. This data gives support to the ‘concept group’ method, not only in the degree of difficulty within a group as reflected by the Percent Correct in Figure 3.14, but also by the consistency of the misconception as seen from the Incorrect Choice distribution. Statements 3 and 4 in

Figure 3.14 present ‘Concept 2’, that greater transverse velocities result in a shorter crossing time, with the vectors in reverse order. Statement 3 reads ‘Time to row across for K is .... for C’, and statement 4 is ‘Time to row across for C is .... for K’. Inspection of the graph indicates the students made the same error, assuming the time to row across for K is less than the time to row across for C, regardless of the manner in which the question was asked. Few students believed the quantities to be equal. In concept group 3, statements 7, 8, 9 and 10, “equal to” is predominantly selected instead of ‘greater than’ or ‘less than’ as appropriate. This detailed feedback makes it easier for the instructor to provide help so that students discover their misconceptions. Finally, as in the previously discussed numerical example, particular hints can be displayed, triggered by the response selected for a statement or by triggered by a combination of responses for several statements.

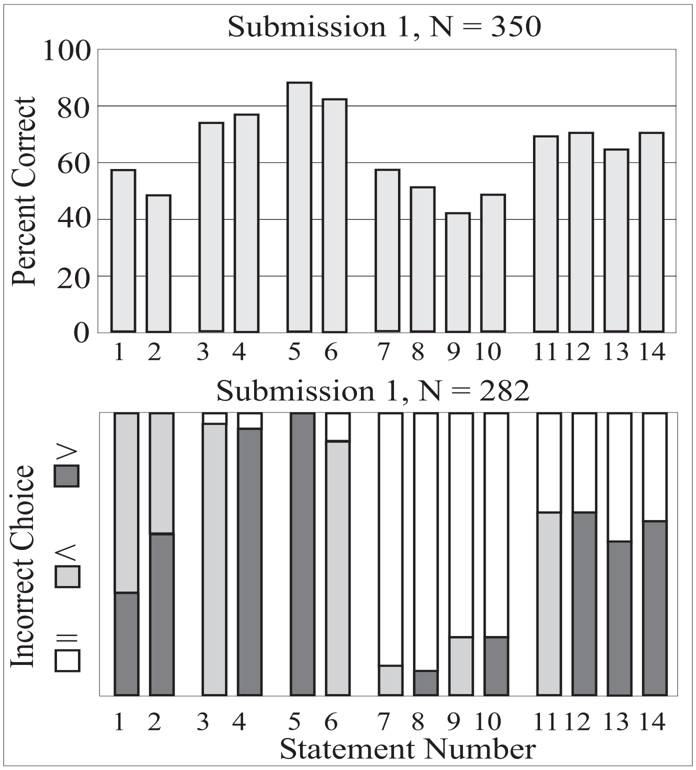


Figure 3.14 Upper Section: Success Rate for Each Possible Statement. Lower Section: Relative distribution of Incorrect Choices, with Dark Gray as "greater than", Light Gray as "Less Than" and Clear as "Equal to"

3.2.4 Homework and Examination Problem Evaluation

The same source code which is used to present problems for on-line homework can also generate them for an on-line examination or for a printed version suitable for a proctored bubble sheet examination which is later machine scored (Albertelli et al., 2003).

LON-CAPA can provide statistical information about every problem in a table (see Table 3.2), which is called "Stats Table". Every part of a *multi-part* problem is distinguished as a separate problem. The *multi-instance* problem is also considered separately, because a particular problem or one part of it might be used in different homework sets. Finally, a table is created which includes all computed information from all students, sorted according to the problem order. In this step, LON-CAPA has provided the following statistical information:

1. **#Stdnts**: Total number of students who take a look at the problem.(Let #Stdnts is equal to n)
2. **Tries:** Total number of submissions to solve the problem ($\sum_{i=1}^{n} x_i$ where x_i denote a student try).
3. **Mod:** Mode, maximum number of submissions for solving the problem.
4. **Mean:** Average number of the submissions. $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
5. **#YES:** Number of students solved the problem correctly.
6. **#yes:** Number of students solved the problem by override.

Sometimes, a student gets a correct answer after talking with the instructor. This type of correct answer is called “corrected by override".

7. **%Wrng:** Percentage of students tried to solve the problem but still incorrect.

$$100 * \left(\frac{n - (\#YES + \#yes)}{n} \right)$$

8. **S.D.:** *Standard Deviation* of the students’ submissions.

$$\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Table 3.2 Statistics table includes general statistics of every problem of the course (Homework Set 1)

Homework Set Order	#Stdnts	Tries	Mod	Mean	#YES	%Wrng	DoDiff	S.D.	Skew.	D.F. 1st	D.F. 2nd
Calculator Skills	256	267	3	1.04	256	0.0	0.04	0.2	5.7	0.03	0.00
Numbers	256	414	17	1.62	255	0.4	0.38	1.6	5.7	0.11	0.02
Speed	256	698	13	2.73	255	0.4	0.63	2.2	1.9	0.06	0.02
Perimeter	256	388	7	1.52	255	0.4	0.34	0.9	2.4	-0.00	0.02
Reduce a Fraction	256	315	4	1.23	256	0.0	0.19	0.5	2.3	0.01	0.00
Calculating with Fractions	256	393	7	1.54	255	0.4	0.35	0.9	2.0	0.15	0.02
Area of a Balloon	254	601	12	2.37	247	2.8	0.59	1.8	1.8	-0.05	-0.02
Volume of a Balloon	252	565	11	2.24	243	3.6	0.57	1.9	2.0	-0.06	-0.03
Units	256	1116	20	4.36	246	3.9	0.78	4.2	1.9	0.18	0.03
Numerical Value of Fraction	256	268	4	1.05	256	0.0	0.04	0.2	3.4	0.01	.00
Vector versus Scalar	254	749	11	2.95	251	1.2	0.66	2.2	1.1	-0.05	-0.05
Adding Vectors	253	1026	20	4.06	250	1.2	0.76	3.6	1.8	0.14	0.00
Proximity	249	663	19	2.66	239	3.6	0.64	2.3	2.8	0.11	-0.10

9. **Skew.:** *Skewness* of the students' submissions.

$$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^3}{(S.D.)^3}=\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^3}{\left(\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2}\right)^3}$$

10. **DoDiff:** *Degree of Difficulty* of the problem.

$$1-\left(\frac{\#YES+\#yes}{\sum_{i=1}^{n}x_i}\right)$$

Clearly, the Degree of Difficulty is always between 0 and 1. This is a useful factor for an instructor to determine whether a problem is difficult, and the degree of this difficulty. Thus, DoDiff of each problem is saved in its meta data.

11. **DoDisc:** *Degree of Discrimination*[4] (or Discrimination Factor) is a standard for evaluating how much a problem discriminates between the upper and the lower students. First, all of the students are sorted according to a criterion. Then, 27% of upper students and 27% lower students are selected from the sorted students applying the mentioned criterion. Finally we obtain the Discrimination Factor from the following difference:

Applied a criterion in 27% upper students - Applied the same Criterion in 27% lower students.

Discrimination Factor is a number in interval [-1,1]. If this number is close to 1, it shows that only upper students have solved this problem. If it is close to 0 it shows that the upper students and the lower students are approximately the same in solving the problem. If this number is negative, it shows that the lower students have more success in solving the problem, and thus this problem is very poor in discriminating the upper and lower students.

We compute the Discrimination Factor from two criteria:

1st Criterion for Sorting the Students: $\dfrac{\sum \text{Partial Credit Awarded}}{\sum_{i=1}^{n} x_i}$

2nd Criterion for Sorting the Students: $\dfrac{\sum (\#YES + \#yes)}{\sum_{i=1}^{n} x_i}$

These measures can also be employed for evaluating resources used in examinations. Examinations as assessment tools are most useful when the content includes a range of difficulty from fairly basic to rather challenging problems. An individual problem within

[4] This name has been given by administration office of Michigan State University for evaluating the exams' problem. Here we expanded this expression to homework problems as well.

an examination can be given a difficulty index (DoDiff) simply by examining the class performance on that problem. Table 3.3 shows an analysis for the first two mid-term examinations in Spring 2004.

Table 3.3 Analysis of Examination Problems (N=393) DoDiff = Difficulty Index DoDisc = Discrimination Index

Problem Number	DoDiff	DoDisc	DoDiff	DoDisc
	Exam 1	Exam 1	Exam 2	Exam 2
1	0.2	0.4	0.7	0.24
2	0.16	0.31	0.13	0.2
3	0.4	0.4	0.19	0.31
4	0.44	0.57	0.41	0.57
5	0.32	0.38	0.52	0.11
6	0	0	0.18	0.26
7	0.23	0.33	0.7	0.36
8	0.21	0.24	0.57	0.35
9	0.36	0.63	0.55	0.58
10	0.4	0.59	0.87	0.14

We can see that Exam 1 was on the average somewhat less difficult than Exam 2. Problem 10 in Exam 2 has DoDisc=0.14 and DoDiff=0.87, indicating it was difficult for all students. The students did not understand the concepts involved well enough to differentiate this problem from a similar problem they had seen earlier. In Exam 1, problems 3, 4, 9, and 10 are not too difficult and nicely discriminating. One striking entry in Table 3.3 is for problem 6 in Exam 1. There both DoDiff and DoDisc are 0. No difficulty and no discrimination together imply a faulty problem. As a result of this situation, a request was submitted to modify LON-CAPA so that in the future an instructor will be warned of such a circumstance.

The distribution of scores on homework assignments differs considerably from that on examinations. This is clearly seen in Figure 3.15.

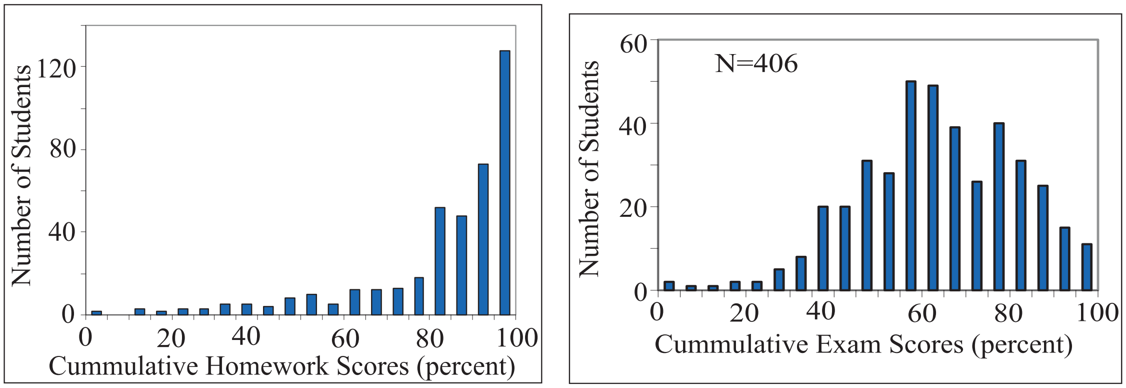


Figure 3.15 Grades on the first seven homework assignments and on the first two midterm examinations

The correlation of homework and examinations is moderate (r=0.43). Students with a good exam score tend to score high on homework but the reverse is not as true. This can be seen in the 3-D plot of the Figure 3.15 data in Figure 3.16. Homework grades peak near 100% as motivated students tend to repeat problems until a correct solution is obtained.

Students also often interpret a high homework grade as indication that they are doing well in the course. To counter that misconception, a readily accessible on-line grade extrapolator provides students a review of their performance to date in the various components of the class, quizzes, mid term exams, and homework. They enter their own estimate of their future performance for the remainder of the semester, as well as for the final examination. This tool then projects a final grade, thus keeping students aware of their progress. As a result of feedback on students' work, those doing very poorly can be identified quite early (Minaei et al., 2003; Thoennessen et al., 1999).

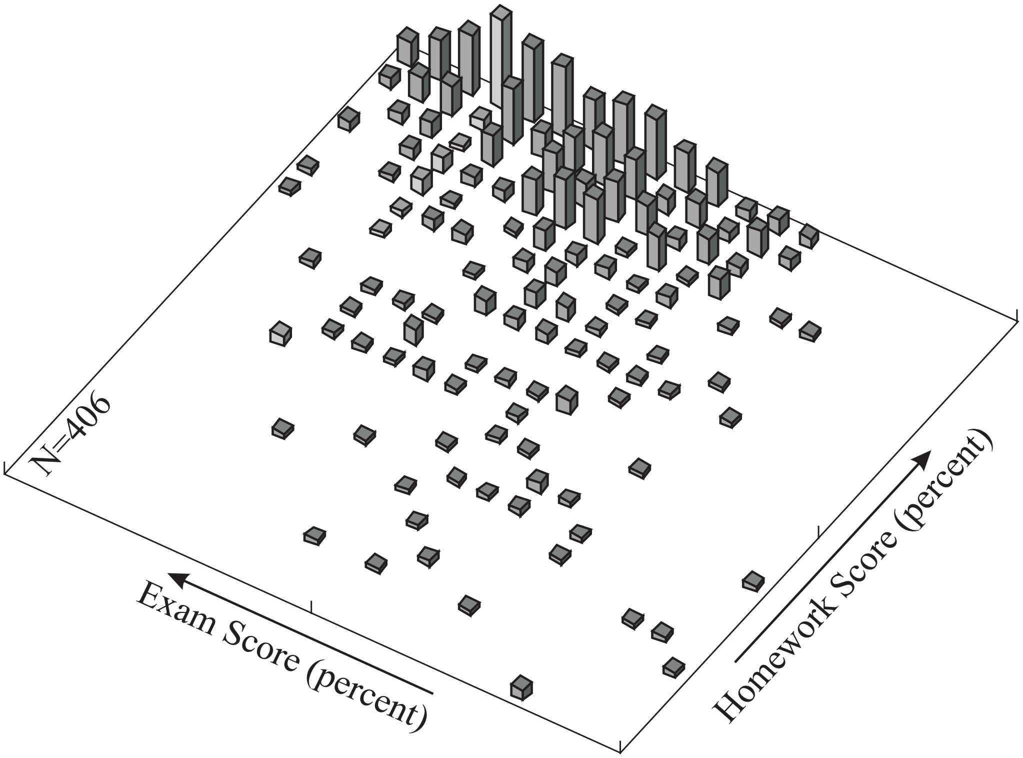


Figure 3.16 Homework vs. Exam Scores. The highest bin has 18 students.

3.3 Summary

LON-CAPA provides instructors or course coordinators full access to the students' educational records. With this access, they are able to evaluate the problems presented in the course after the students have used the educational materials, through some statistical reports. LON-CAPA also provides a quick review of students' submissions for every problem in a course. The instructor may monitor the number of submissions of every student in any homework set and its problems. The total numbers of solved problems in a homework set as compared with the total number of solved problems in a course are represented for every individual student.

LON-CAPA reports a large volume of statistical information for every problem e.g., "total number of students who open the problem," "total number of submissions for the problem," "maximum number of submissions for the problem," "average number of submissions per problem," "number of students solving the problem correctly," etc. This information can be used to evaluate course problems as well as the students. More details can be found in Albertelli et al. (2002) and Hall et al. (2004). Aside from these evaluations, another valuable use of data will be discussed in the next chapter.

Chapter 4 Predicting Student Performance

The objective in this chapter is to *predict* the students' final grades based on the features which are extracted from their (and others') homework data. We design, implement, and evaluate a series of pattern classifiers with various parameters in order to compare their performance in a real data set from the LON-CAPA system. This experiment provides an opportunity to study how pattern recognition and classification theory could be put into practice based on the logged data in LON-CAPA. The error rate of the decision rules is tested on one of the LON-CAPA data sets in order to compare the performance accuracy of each experiment. Results of individual classifiers, and their combination, as well as error estimates, are presented.

The problem is whether we can find the good features for *classifying* students! If so, we would be able to identify a *predictor* for any individual student after doing a couple of homework sets. With this information, we would be able to *help* a student use the resources better.

The difficult phase of the experiment is properly pre-processing and preparing the data for classification. Some Perl modules were developed to extract and segment the data from the logged database and represent the useful data in some statistical tables and graphical charts. More details of these tasks have been explained in a part of previous chapter which is dedicated for data acquisition and data representation.

4.1 Data set and Class Labels

As the first step in our study, in order to have an experiment in *student classification*, we selected the student and course data of a LON-CAPA course, PHY183 (Physics for Scientists and Engineers I), which was held at MSU in spring semester 2002. Then we extend this study to more courses. This course integrated 12 homework sets including 184 problems. About 261 students used LON-CAPA for this course. Some of the students dropped the course after doing a couple of homework sets, so they do not have any final grades. After removing those students, 227 valid samples remained. You can see the grade distribution of the students in the following chart (Figure 4.1)

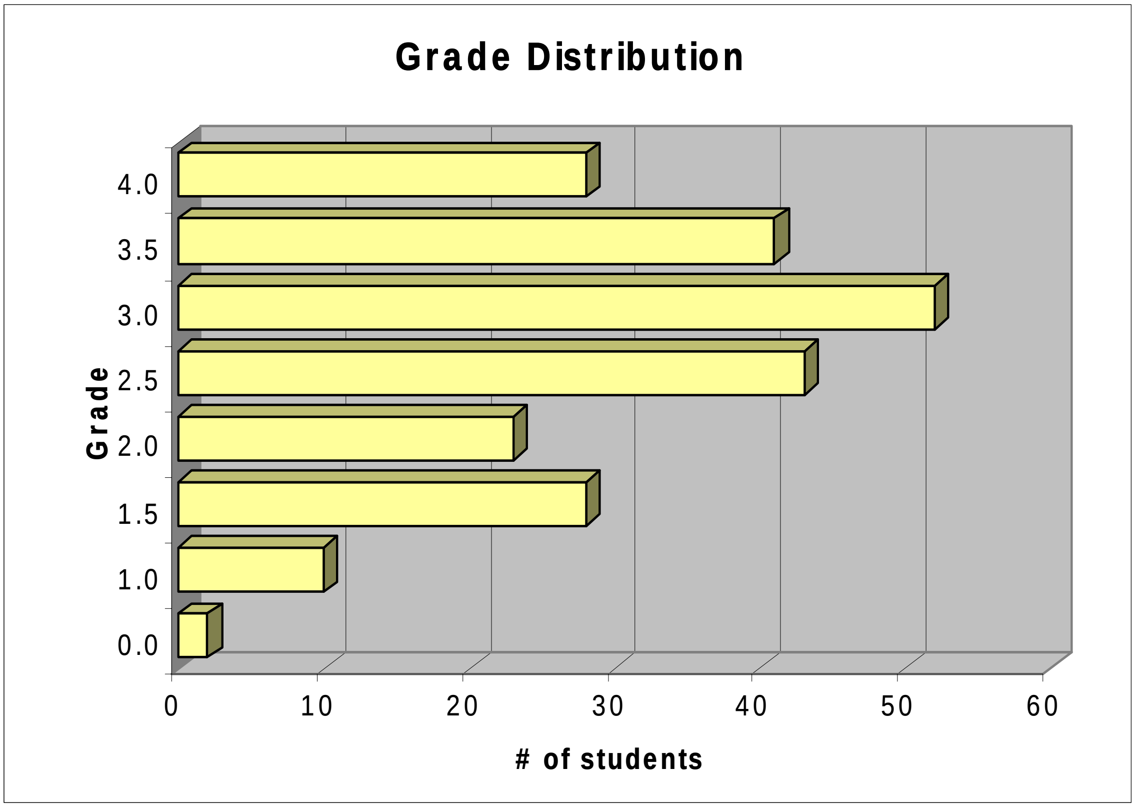


Figure 4.1 Graph of distribution of grades in course PHY183 SS02

We can group the students regarding their final grades in several ways, 3 of which are:

1. The 9 possible class labels can be the same as students' grades, as shown in Table 4.1.
2. We can group them into three classes, "*high*" representing grades from 3.5 to 4.0, "*middle*" representing grades from 2.5 to 3, and "*low*" representing grades less than 2.5, as shown in table 4.2.
3. We can also categorize students with one of two class labels: "*Passed*" for grades above 2.0, and "*Failed*" for grades less than or equal to 2.0, as shown in table 4.3.

Table 4.1. 9-Class labels regarding students' grades in course PHY183_ SS02

Class	Grade	# of Student	Percentage
1	0.0	2	0.9%
2	0.5	0	0.0%
3	1.0	10	4.4%
4	1.5	28	12.4%
5	2.0	23	10.1%
6	2.5	43	18.9%
7	3.0	52	22.9%
8	3.5	41	18.0%
9	4.0	28	12.4%

Table 4.2. 3-Class labels regarding students' grades in course PHY183 SS02

Class	Grade	Student #	Percentage
High	Grade >= 3.5	69	30.40%
Middle	2.0 < Grade < 3.5	95	41.80%
Low	Grade <= 2.0	63	27.80%

Table 4.3. 2-class labels regarding students' grades in course PHY183 SS02

Class	Grade	Student #	Percentage
Passed	Grade > 2.0	164	72.2%
Failed	Grade <= 2.0	63	27.80%

We can predict that the error rate in the first class grouping should be higher than the others, because the sample size among the 9-Classes differs considerably.

The present classification experiment focuses on the first six extracted students' features based on the PHY183 Spring 2002 class data.

1. Total number of correct answers. (Success rate)
2. Getting the problem right on the first try, vs. those with high number of submissions. (Success at the first try)
3. Total number of attempts before final answer is derived
4. Total time that passed from the first attempt, until the correct solution was demonstrated, regardless of the time spent logged in to the system. Also, the time at which the student got the problem correct relative to the due date. Usually better students get the homework completed earlier.
5. Total time spent on the problem regardless of whether they got the correct answer or not. Total time that passed from the first attempt through subsequent attempts until the last submission was demonstrated.
6. Participating in the communication mechanisms, vs. those working alone. LON-CAPA provides online interaction both with other students and with the instructor.

4.2 Classifiers

Pattern recognition has a wide variety of applications in many different fields; therefore it is not possible to come up with a single classifier that can give optimal results in each case. The optimal classifier in every case is highly dependent on the problem domain. In practice, one might come across a case where no single classifier can perform at an acceptable level of accuracy. In such cases it would be better to pool the results of different classifiers to achieve the optimal accuracy. Every classifier operates well on different aspects of the training or test feature vector. As a result, assuming appropriate conditions, combining multiple classifiers may improve classification performance when compared with any single classifier.

4.2.1 Non-tree based classifiers

We compare some popular non-parametric pattern classifiers and a single parametric pattern classifier according to their error estimates. Six different classifiers over one of the LON-CAPA data sets are compared. The classifiers used include *Quadratic Bayesian classifier, 1-nearest neighbor (1-NN)*, *k-nearest neighbor (k-NN)*, *Parzen-window, multi-layer perceptron (MLP),* and *Decision Tree.*[5] These classifiers are some of the most common classifiers used in practical classification problems. After some preprocessing operations were made on the data set, the error rate of each classifier is reported. Finally, to improve performance, a combination of classifiers is presented.

[5] The first five classifiers are coded in MATLAB^{TM} 6.0, and for the decision tree classifiers we have use some available software packages such as C5.0, CART, QUEST, CRUISE. We will discuss the Decision Tree-based software in the next section. In this section we deal with non-tree classifiers.

4.2.1.1 Combination of Multiple Classifiers (CMC)

In combining multiple classifiers we seek to improve classification accuracy. There are different ways one can think of combining classifiers:

- The simplest way is to find the overall error rate of the classifiers and choose the one which has the lowest error rate for the given data set. This is called an *offline CMC*. This may not really seem to be a CMC; however, in general, it has a better performance than individual classifiers. The output of this combination will simply be the best performance in each column in Figures 4.3 and 4.5.
- The second method, which is called *online CMC*, uses all the classifiers followed by a vote. The class getting *maximum votes* from the individual classifiers will be assigned to the test sample. This method seems, intuitively, to be better than the previous one. However, when we actually tried this on some cases of our data set, the results were not more accurate than the best result from the previous method. Therefore, we changed the rule of majority vote from "*getting more than 50% of the votes*" to "*getting more than 75% of the votes*". We then noticed a significant improvement over offline CMC. Table 4.6 shows the actual performance of the individual classifier and online CMC over our data set.
- Woods et al. (1995) suggest a third method, which is called DSC-LA (Dynamic Selection of Classifiers based on the Local Accuracy estimates). This method takes a particular test sample, investigates the local

neighborhood of that sample using all the individual classifiers and the one which performs best is chosen for the decision-making[6].

Besides CMC, we also show the outcomes for an “Oracle” which chooses the correct results if *any* of the classifiers classified correctly, as Woods et al. (1995) has presented in their article.

4.2.1.2 Normalization

Having assumed in Bayesian and Parzen-window classifiers that the features are normally distributed, it is necessary that the data for each feature be normalized. This ensures that each feature has the same weight in the decision process. Assuming that the given data is conforms to a Gaussian distribution; this normalization is performed using the mean and standard deviation of the training data. In order to normalize the training data, it is necessary first to calculate the sample mean, μ, and the standard deviation, σ, of each feature (column) in the data set, and then normalize the data using the following equation:

$$x_i = \frac{x_i - \mu}{\sigma} \tag{4.1}$$

This ensures that each feature of the training data set has a normal distribution with a mean of zero and a standard deviation of one. In addition, the *k*NN method requires normalization of all features into the same range. However, we should be cautious in using the normalization before considering its effect on classifiers’ performances. Table

[6] We have not implemented this method in this proposal yet.

4.4 shows a comparison of Error Rate and Standard Deviation, using the classifiers in both normalized and un-normalized data in the case of 3 classes.

Table 4.4 Comparing Error Rate of classifiers with and without normalization in the case of 3 classes

3-Classes	With Normalization		Without Normalization	
Classifier	**Error rate**	**S.D**	**Error rate**	**S. D.**
Bayes	0.4924	0.0747	0.5528	0.0374
1NN	0.5220	0.0344	0.5864	0.041
KNN	0.5144	0.0436	0.5856	0.0491
Parzen	0.5096	0.0408	0.728	0
MLP	0.4524	0.0285	0.624	0
CMC	0.2976	0.0399	0.3872	0.0346
Oracle	0.1088	0.0323	0.1648	0.0224

Thus, we tried the classifiers with and without normalization. Table 4.4 clearly shows a significant improvement in most classification results after normalization. Here we have two findings:

1. The Parzen-Window classifier and MLP do not work properly without normalizing the data. Therefore, we have to normalize data when using these two classifiers.
2. Decision tree classifiers do not show any improvement on their classification performance after normalization, so we ignore it in using tree classifiers. We will study the decision tree classifier later, though the Decision Tree classifiers' results are not introduced in Table 4.4.

4.2.1.3 Comparing 2-fold and 10-fold Cross-Validation

In *k*-fold cross-validation, we divide the data into k subsets of approximately equal size. We train the data *k* times, each time leaving out one of the subsets from training, but using only the omitted subset to compute the error threshold of interest. If *k* equals the sample size, this is called "*Leave-One-Out*" cross-validation. (Duda et al. 2001; Kohavi, 1995). Leave-One-Out cross-validation provides an almost unbiased estimate of true accuracy, though at a significant computational cost. In this proposal both 2-fold and 10-fold cross validation are used.

In ***2-fold*** cross-validation, the order of the observations, both training and test, are randomized before every trial of every classifier. Next, every sample is divided amongst the test and training data, with 50% going to training, and the other 50% going to test. This means that testing is completely independent, as no data or information is shared between the two sets. At this point, we classify[7] the test sets after the training phase of all the classifiers. We repeat this *random cross validation* ten times for all classifiers.

In ***10-fold*** cross-validation, the available data (here, the 227 students' data) are divided into 10 blocks containing roughly equal numbers of cases and class-value distributions. For each block (10% of data) in turn, a model is developed using the data in the remaining blocks (90% of data, *the training set*), and then it is evaluated on the cases in the hold-out block (*the test set*). When all the tests (10 tests) are completed, each sample in the data will have been used to test the model exactly once. The average performance on the tests is then used to predict the true accuracy of the model developed from all the data. For *k*-values of 10 or more, this estimate is more reliable and is much more accurate than a re-substitution estimate.

[7] All the code was written in MATLAB^{TM} 6.5

Table 4.5 Comparing Error Rate of classifiers 2-fold and 10-fold Cross-Validation in the case of 3 classes

3-Classes	10-fold Cross-Validation		2-fold Cross-Validation	
Classifier	**Error Rate**	**S.D.**	**Error Rate**	**S.D.**
Bayes	0.5	0.0899	0.5536	0.0219
1NN	0.4957	0.0686	0.5832	0.0555
KNN	0.5174	0.0806	0.576	0.0377
Parzen	0.5391	0.085	0.4992	0.036
MLP	0.4304	0.0806	0.4512	0.0346
CMC	0.313	0.084	0.3224	0.0354
Oracle	0.1957	0.0552	0.1456	0.0462

Table 4.5 shows comparison of Error Rate and Standard Deviation using the classifiers in both 2-fold and 10-fold cross-validation in the case of the 3-Classes. You can see that the 10-fold cross-validation in relation to individual classifier has slightly more accurate than 2-fold cross validation, but in relation to combination of classifiers (CMC) there is no a significant difference. Nonetheless, we selected 10-fold cross validation for error estimation in this proposal.

4.2.1.4 Results, Error Estimation

The experimental results were averaged and are presented in the charts below. They show the effect of selecting the data randomly on the average error rate. The average error rate and its standard deviation, which is associated with each classifier, is shown in the tables as well as the chart. Each table summarizes the results of all the classifiers on our data set.

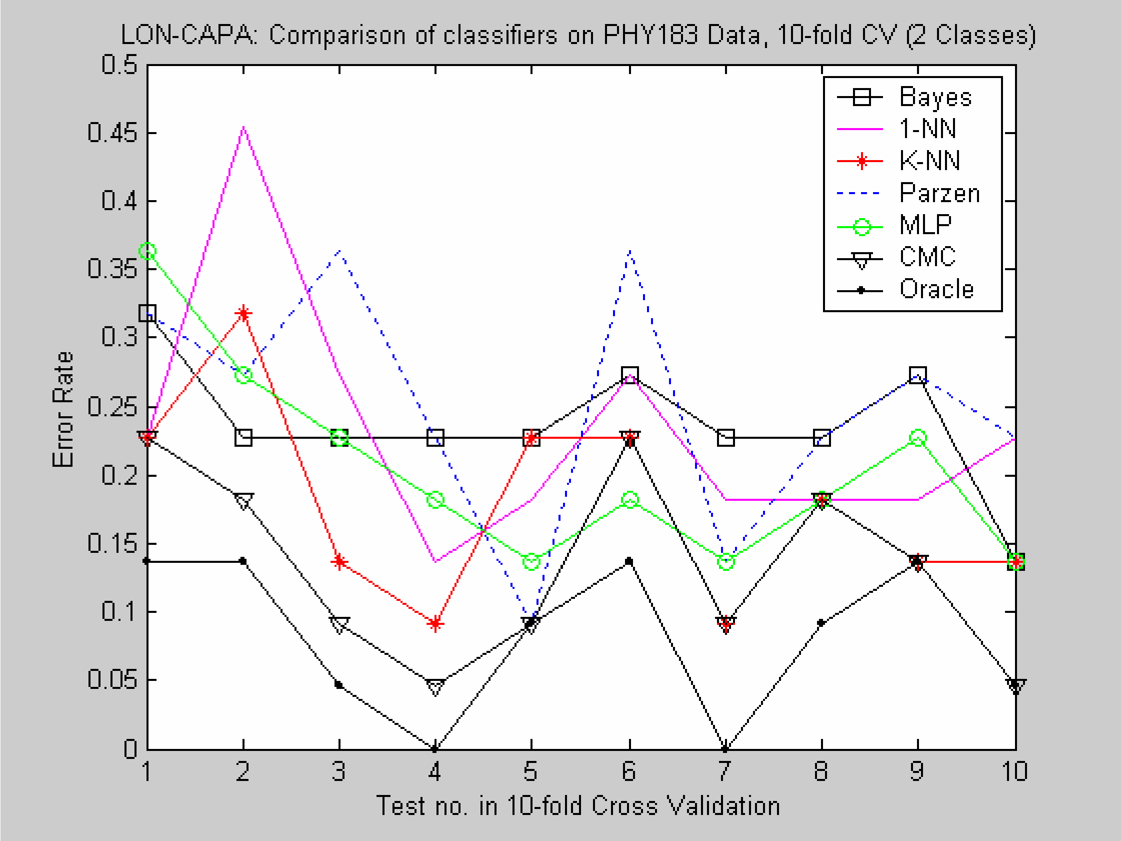
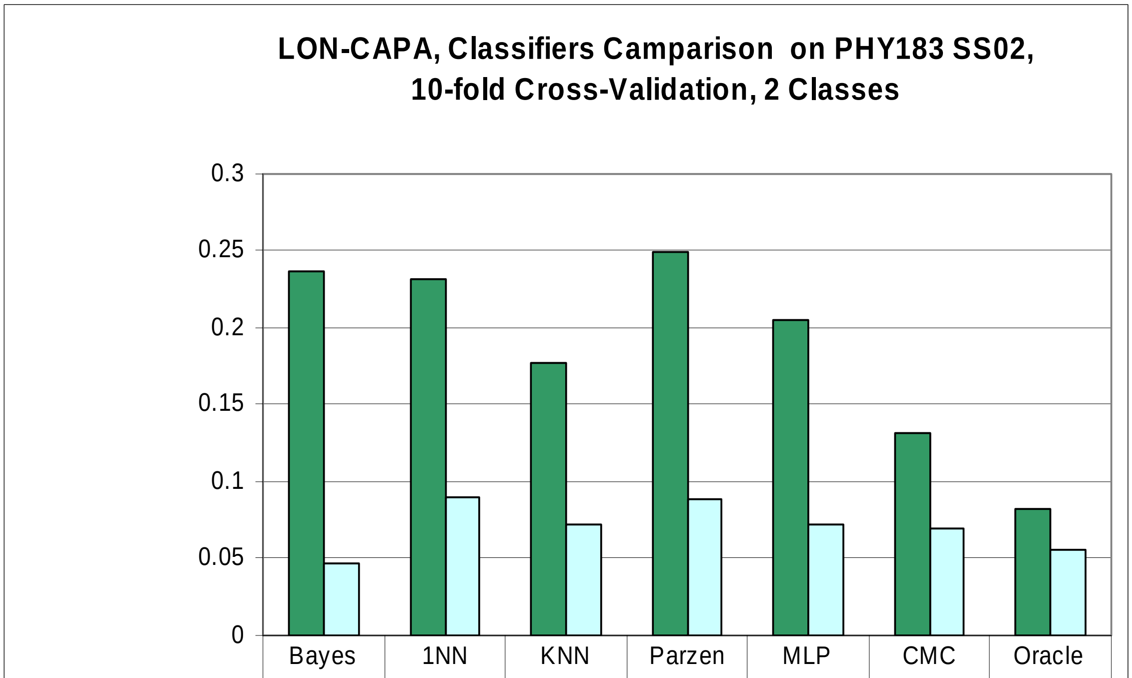


Figure 4.2: Comparing Error Rate of classifiers with 10-fold Cross-Validation in the case of 2-Classes

The standard deviation of error rate shows the variance of the error rate during cross validation. The error rate is measured in each round of cross validation by:

$$\text{Error Rate in each round} = \frac{\text{Total missclassified of test examples}}{\text{Total number of test examples}}$$

After 10 rounds, the average error rate and its standard deviation are computed and then plotted. This metric was chosen due to its ease of computation and intuitive nature. Figure 4.2 and 4.3 show the comparison of classifiers' error rate when we classify the students into two categories, "*Passed*" and "*Failed*". The best performance is for *k*NN with 82% accuracy, and the worst classifier is Parzen-window with 75% accuracy. CMC in the case of 2-Classes classification has 87% accuracy.



	Bayes	1NN	KNN	Parzen	MLP	CMC	Oracle
🟩 Average Error Rate	0.2364	0.2318	0.1773	0.25	0.2045	0.1318	0.0818
□ Standard Deviation	0.0469	0.0895	0.0725	0.089	0.0719	0.0693	0.0559

Figure 4.3: Table and graph to compare Classifiers' Error Rate, 10-fold CV in the case of 2-Classes

It is noticeable that these processes were done after we had found the optimal *k* in the *k*NN algorithm and after we had tuned the parameters in MLP and after we had found the optimal *h* in the Parzen-window algorithm. Finding the best *k* for *k*NN is not difficult, and its performance is the best in the case of 2-Classes, though is not as good as the other classifiers in the case of 3-Classes, as is shown in Figure 4.4.

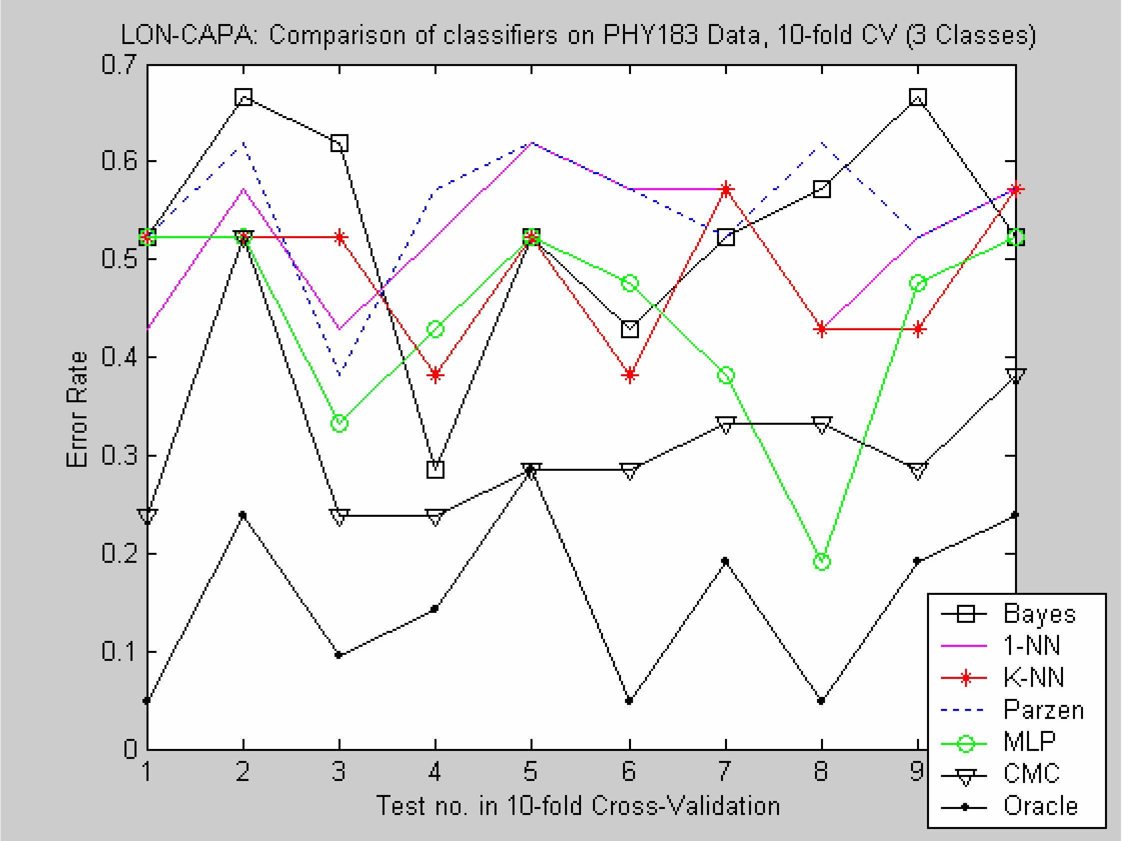
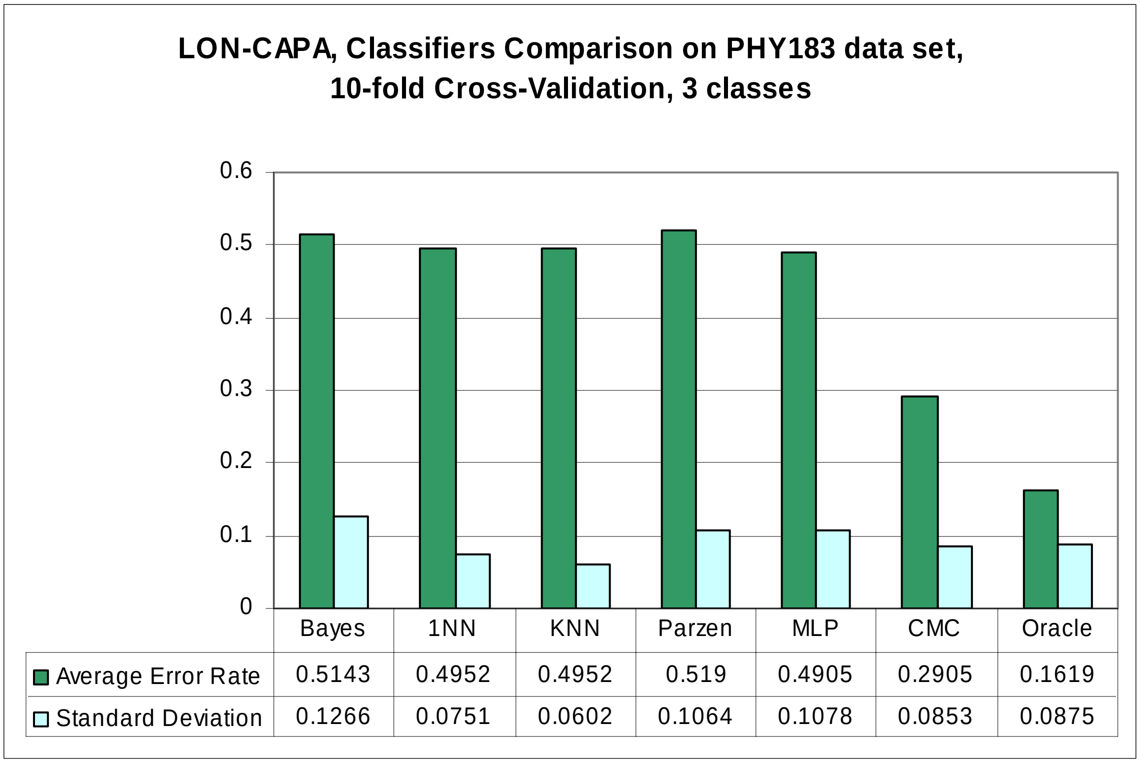


Figure 4.4: Comparing Error Rate of classifiers with 10-fold Cross-Validation in the case of 3-Classes

Working with Parzen-window classifier is not as easy because finding the best width for its window is not straitforward. The MLP classifier is the most difficult classifier to work with. Many parameters have to be set properly to make it work optimally. For example, after many trials and errors we found that the structure of the network in the case of 3-classes, the 4-3-3 (one hidden layer with 3 neurons in hidden layer) works better, and in the case of 2-classes, if we have 2 hidden layer with 2 or 4 neurons in each hidden layer, would lead to a better performance. There is no algorithm to set the number of epochs and learning rates in the MLP. However, sometimes MLP has the best performance in our data set. As shown in the Table 4.6, MLP is slightly better than the other individual classifier. In the case of 9-Classes we could not set the MLP to work properly, so we have not brought the result of MLP classifier into the final result in table 4.6.



	Bayes	1NN	KNN	Parzen	MLP	CMC	Oracle
🟩 Average Error Rate	0.5143	0.4952	0.4952	0.519	0.4905	0.2905	0.1619
□ Standard Deviation	0.1266	0.0751	0.0602	0.1064	0.1078	0.0853	0.0875

Figure 4.5 Comparing Classifiers' Error Rate, 10-fold CV in the case of 3-Classes

As predicted before, the error rate in the case of 9-Classes is much higher than in other cases. The final results of the five classifiers and their combination in the case of 2-Classes, 3-Classes, and 9-Classes are shown in Table 4.6.

In the case of 9-Classes, 1-NN works better than the other classifiers. Final results in Table 4.6 show that CMC is the most accurate classifier compared to individual classifiers. In the case of 2-Classes it improved by **5**%, in the case of 3-Classes it improved by **20**%, and in the case of 9-Classes it improved by **22**%, all in relation to the best individual classifiers in the corresponding cases.

Table 4.6: Comparing the Performance of classifiers, in all cases: 2-Classes, 3-Classess, and 9-Classes, Using 10-fold Cross-Validation in all cases.

	Error Rate		
Classifier	**2-Classes**	**3-Classes**	**9-Classes**
Bayes	0.2364	0.5143	0.77
1NN	0.2318	0.4952	0.71
KNN	0.1773	0.4952	0.725
Parzen	0.25	0.519	0.795
MLP	0.2045	0.4905	-
CMC	0.1318	0.2905	0.49
Oracle	0.0818	0.1619	-

One important finding is that when our individual classifiers are working well and each has a high level of accuracy; the benefit of combining classifiers is small. Thus, CMC has little improvement in classification performance while it has a significant improvement in accuracy when we have *weak learner*[8] classifiers.

We tried to improve the classification efficiency by stratifying the problems in relation to their *degree of difficulty.* By choosing some specific conceptual subsets of the students' data, we did not achieve a significant increase in accuracy with this parameter. In the next section, we explain the results of decision tree classifiers on our data set, while also discussing the relative importance of student-features and the correlation of these features with category labels.

4.2.2 Decision Tree-based software

Decision trees have proved to be valuable tools for the description, classification and generalization of data. Many users find decision trees easy to use and understand. As a result, users more easily trust decision tree models than they do "black box" models, such

[8] "*Weak learner*" means that the classifier has accuracy only slightly better than chance (Duda et al., 2001)

as models produced by neural networks. Many tools and software have been developed to implement decision tree classification. Lim et al. (2000) has an insightful study about comparison of prediction accuracy, complexity, and training time of thirty-three classification algorithms; twenty-two decision trees, nine statistical and two neural network algorithms are compared on thirty-two data sets in terms of classification accuracy, training time, and (in the case of trees) number of leaves. In this proposal we used C5.0, CART, QUEST, and CRUISE software to test tree-based classification. Some statistical software is employed for multiple linear regression on our data set. First we have a brief view of the capabilities, features and requirements of these software packages. Then we gather some of the results and compare their accuracy to non-tree based classifiers.

4.2.2.1 C5.0

Decision tree learning algorithms, for example, ID3, C5.0 and ASSISTANT (Cestnik et al., 1987), search a completely expressive hypothesis space and are used to approximate discrete valued target functions represented by a decision tree. In our experiments the C5.0 inductive learning decision tree algorithm was used. This is a revised version[9] of C4.5 and ID3 (Quinlan 1986, 1993) and includes a number of additional features. For example, the *Boosting* option causes a number of classifiers to be constructed - when a case is classified, all of these classifiers are consulted before making a decision. Boosting will often give a higher predictive accuracy at the expense of

[9] It is the commercial version of the C4.5 decision tree algorithm developed by Ross Quinlan. See5/C5.0 classifiers are expressed as decision trees or sets of if-then rules. RuleQuest provides C source code so that classifiers constructed by See5/C5.0 can be embedded in your own systems.

increased classifier construction time. For our experiments, however, data set boosting was not found to improve prediction accuracy.

When a continuous feature is tested in a decision tree, there are branches corresponding to the conditions: "*Feature Value* ≤ *Threshold"* and "*Feature Value* > *Threshold,*" for some threshold chosen by C5.0. As a result, small movements in the feature value near the threshold can change the branch taken from the test. There have been many methods proposed to deal with continuous features (Quinlan, 1988; Chan et al., 1992; Ching et al., 1995). An option available in C5.0 uses *fuzzy thresholds* to soften this knife-edge behavior for decision trees by constructing an interval close to the threshold. This interval plays the role of margin in neural network algorithms. Within this interval, both branches of the tree are explored and the results combined to give a predicted class.

Decision trees constructed by C5.0 are post pruned before being presented to the user. The "*Pruning Certainty Factor"* governs the extent of this simplification. A higher value produces more elaborate decision trees and rule sets, while a lower value causes more extensive simplification. In our experiment a certainty factor of 25% was used. If we change the certainty factor, we may obtain different results.

C5.0 needs four types of files for generating the decision tree for a given data set, out of which two files are optional:

The first file is the *.names* file. It describes the attributes and classes. The first line of the *.names* file gives the classes, either by naming a discrete attribute (the *target* attribute) that contains the class value, or by listing them explicitly. The attributes are then defined in the order that they will be given for each case. The attributes can be either explicitly or implicitly defined. The value of an explicitly defined attribute is given directly in the data. The value of an implicitly-defined attribute is specified by a formula. In our case, data attributes are explicitly defined.

The second file is the *.data* file. It provides information on the *training* cases from which C5.0 will extract patterns. The entry for each case consists of one or more lines that give the values for all explicitly defined attributes. The '?' is used to denote a value that is missing or unknown. Our data set had no missing features. Also, 'N/A' denotes a value that is not applicable for a particular case.

The third file used by C5.0 consists of new test cases on which the classifier can be evaluated and is the *.test* file. This file is optional and, if used, has exactly the same format as the *.data* file. We gave a *.test* file for our data set.

The last file is the *.costs* file. In applications with differential misclassification costs, it is sometimes desirable to see what affect costs have on the construction of the classifier. In our case all misclassification costs were the same so this option was not implemented.

After the program was executed on the PHY183 SS02 data set we obtained results for both the training and testing data. A confusion matrix was generated in order to show the misclassifications. The confusion matrices for three types of classification in our data set that are, 2-Classes, 3-Classes and 9-Classes are in Appendix A. You can also find some rule set samples resulted from the rule-set option in C5.0, as well as a part sample of the tree produced by C5.0 in Appendix A.

Using 10-fold cross validation we got **79.3%** accuracy in 2-Classes, **56.8%** accuracy in 3-Classes, **25.6%** accuracy in 9-Classes.

One of the important and exciting experiments of C5.0 is to use a training, and test set; and thus at least a 10-fold cross-validation to get a desirable result. For example, in the case of 3-Classes, we might get approximately 75% accuracy in the training set, while boosting might improve accuracy up to 90% or 95%. Unfortunately, this is *overfitting*, or *overtraning*, and so we therefore would not be able to generalize these results or this complex training model to test the unseen data because of overfitting.

4.2.2.2 CART

CART[10] uses an exhaustive search method to identify useful tree structures of data. It can be applied to any data set and can proceed without parameter setting. Comparing CART analyses with stepwise logistic regressions or discriminant analysis, CART typically performs better on the learning sample. Listed below are some technical aspects of CART:

CART is a nonparametric procedure and does not require specification of a functional form. CART uses a stepwise method to determine splitting rules, and thus no advance selection of variables is necessary, although certain variables such as ID numbers and reformulations of the dependent variable should be excluded from the analysis. Also, CART's performance can be enhanced by proper feature selection and

[10] CART$^{(tm)}$ (Classification And Regression Trees) is a data mining tool exclusively licensed to Salford Systems (http://www.salford-systems.com). CART is the implementation of the original program by Breiman, Friedman, Olshen, and Stone. We used CART version 5.02 under windows for our classification. Using CART we are able to get many interesting textual and graphical reports, some of which are presented in Appendix A. It is noticeable that CART does not use any description files to work with. Data could be read as a text file or any popular database or spreadsheet.

creation of predictor variables. There is no need to experiment with monotone transformations of the independent variables, such as logarithms, square roots or squares. In CART, creating such variables will not affect the resulting trees unless linear combination splits are used. Outliers among the independent variables generally do not affect CART because splits usually occur at non-outlier values. Outliers in the dependent variable are often separated into nodes where they no longer affect the rest of the tree. CART does not require any preprocessing of the data. In particular, continuous variables do not have to be recoded into discrete variable versions prior to analysis. While the CART default is to split nodes on single variables, it will optionally use linear combinations of non-categorical variables. For each split in the tree, CART develops alternative splits (surrogates), which can be used to classify an object when the primary splitting variable is missing. Thus, CART can be used effectively with data that has a large fraction of missing values.

One of the advantages of CART is presenting the importance of independent variables in predicting both classification mode and regression mode. Each variable in the CART tree has an importance score based on how often and with what significance it served as primary or surrogate splitter throughout the tree. The scores reflect the contribution each variable makes in classifying or predicting the target variable, with the contribution stemming from both the variable's role in primary splits and its role as a surrogate splitter (Dan and Colla, 1998). In Table 4.7 and 4.8, the importance of the six features (independent variables) are scored in the case of 2-classes with Gini splitting criterion and 3-classes with Entropy splitting criterion respectively.

Table 4.7 Variable (feature) Importance in 2-Classes Using Gini Criterion

<table>
<tr><th>Variable</th><th></th><th></th></tr>
<tr><td>TOTCORR</td><td>100.00</td><td>||||||||||||||||||||||||||||||||||||||||||</td></tr>
<tr><td>TRIES</td><td>56.32</td><td>|||||||||||||||||||||||</td></tr>
<tr><td>FIRSTCRR</td><td>4.58</td><td>|</td></tr>
<tr><td>TOTTIME</td><td>0.91</td><td></td></tr>
<tr><td>SLVDTIME</td><td>0.83</td><td></td></tr>
<tr><td>DISCUSS</td><td>0.00</td><td></td></tr>
</table>

Table 4.8 Variable (feature) Importance in 2-Classes, Using Entropy Criterion

<table>
<tr><th>Variable</th><th></th><th></th></tr>
<tr><td>TOTCORR</td><td>100.00</td><td>||||||||||||||||||||||||||||||||||||||||||</td></tr>
<tr><td>TRIES</td><td>58.61</td><td>||||||||||||||||||||||||</td></tr>
<tr><td>FIRSTCRR</td><td>27.70</td><td>|||||||||||</td></tr>
<tr><td>SLVDTIME</td><td>24.60</td><td>||||||||||</td></tr>
<tr><td>TOTTIME</td><td>24.47</td><td>||||||||||</td></tr>
<tr><td>DISCUSS</td><td>9.21</td><td>|||</td></tr>
</table>

The results in Tables 4.7 and 4.8 show that the most important feature (which has the highest correlation with the predicted variables) is the "*Total number of Correct answers,*" and the least useful variable is the "*Number of Discussions.*" If we consider the economical aspect of computation cost, we can remove the less important features.

In our experiment, we used both 10-fold Cross-Validation and Leave-One-Out method. We found that the error rates in training sets are not improved in the case of 2-Classes and 3-Classes, but the misclassifications in the test sets are improved when we switch from 10-fold Cross-Validation to Leave-One-Out. Yet, in the case of 9-Classes both training and testing sets are improved significantly when switching occurs, as is shown in Table 4.9 and 4.10. How can we interpret improvement variation between classes?

Table 4.9: Comparing the Error Rate in CART, using 10-fold Cross-Validation in learning and testing set.

Splitting Criterion	2-Classes		3-Classes		9-Classes	
	Training	Testing	Training	Testing	Training	Testing
Gini	17.2%	19.4%	35.2%	48.0%	66.0%	74.5%
Symmetric Gini	17.2%	19.4%	35.2%	48.0%	66.0%	74.5%
Entropy	18.9%	19.8%	37.9%	52.0%	68.7%	76.2%
Twoing	17.2%	19.4%	31.3%	47.6%	54.6%	75.3%
Ordered Twoing	17.2%	20.7%	31.7%	48.0%	68.3%	74.9%

Table 4.10: Comparing the Error Rate in CART, using Leave-One-Out method in learning and testing test.

Splitting Criterion	2-Classes		3-Classes		9-Classes	
	Training	Testing	Training	Testing	Training	Testing
Gini	17.2%	18.5%	36.6%	41.0%	46.7%	66.9%
Symmetric Gini	17.2%	18.5%	36.6%	41.0%	46.7%	66.9%
Entropy	17.2%	18.9%	35.2%	41.4%	48.0%	69.6%
Twoing	17.2%	18.5%	38.3%	40.1%	47.1%	68.7%
Ordered Twoing	18.9%	19.8%	35.2%	40.4%	33.9%	70.9%

Discussion of improvement variation:

In the case of 2-Classes, there is no improvement in the training phase and a slight (1%) improvement in the test phase. It shows that we can have a more reliable model with the Leave-One-Out method for student classification. It shows that the model we obtained in training phase is approximately complete.

In the case of 3-Classes, when we switch from 10-fold to Leave-One-Out the results in the training phase become slightly worse, but we achieve approximately 7.5% improvement. It shows that our model was not complete in 10-fold for predicting the unseen data. Therefore, it is better to use Leave-One-Out to get a more complete model for classifying the students into three categories.

In the case of 9-Classes when we switch from 10-fold to Leave-One-Out, the results in both training and test sets improve significantly. However, we cannot conclude that our new model is complete. This is because the big difference between the results in training and testing phase shows that our model is suffering from *overfitting.* It means that our training samples are not enough to construct a complete model for predicting the category labels correctly; more data is required to come to an adequate solution.

After discussing the CART results it is worth noting that by using CART we are able to produce many useful textual and graphical reports, some of which are presented in Appendix A. In next section we discuss the other decision tree software results. One of the advantages of CART is that it does not require any description files; therefore data can be read as a text file or any popular database or spreadsheet file format.

4.2.2.3 QUEST, CRUISE[11]

QUEST is a statistical decision tree algorithm for classification and data mining. The objective of QUEST is similar to that of the algorithm used in CART and described in Breiman, et al. (1984). The advantages of QUEST are its *unbiased* variable selection technique by default, its use of imputation instead of surrogate splits to deal with missing values, and its ability to handle categorical predictor variables with many categories. If there are no missing values in the data, QUEST can use the CART greedy search algorithm to produce a tree with univariate splits.

[11] **QUEST** (Quick, Unbiased and Efficient Statistical Tree) A classification tree restricted to binary splits
CRUISE (Classification Rule with Unbiased Interaction Selection and Estimation) A classification tree that splits each node into two or more sub-nodes. These new software were developed by Wei-Yin Loh at the University of Wisconsin-Madison, Shih at University of Taiwan, and Hyunjoong Kim at University of Tennessee,. vastly-improved descendant of an older algorithm called FACT.

QUEST needs two text input files: **1) Data file:** This file contains the training samples. Each sample consists of observations on the class (or response or dependent) variable and the predictor (or independent) variables. The entries in each sample record should be comma or space delimited. Each record can occupy one or more lines in the file, but each record must begin on a new line. Record values can be numerical or character strings. Categorical variables can be given numerical or character values. **2) Description file:** This file is used to provide information to the program about the name of the data file, the names and the column locations of the variables, and their roles in the analysis.

The following is the description file:

phy183.dat

"?"

```
column, var, type
     1  1stGotCrr  n
     2  TotCorr  n
     3  AvgTries  n
     4  TimeCorr  n
     5  TimeSpent  n
     6  Discuss  n
     7  Class2  x
     8  Class3  d
     9  Class9  x
     10  Grade  x
```

In the first line, we put the name of data file (`phy183.dat`), and in the second line we put the character used to denote missing data (`?`). In our data set we have no missing data. The position (`column`), name (`var`) and role (`type`) of each variable follow, with one line for each variable. The following roles for the variables are permitted: "**c**" stands for categorical variable; "**d**" for class (dependent) variable; only one variable can have the `d` indicator; "**n**" for a numerical variable; and "**x**" which indicates that the variable is excluded from the analysis.

QUEST allows both interactive and batch mode. By default it uses “discriminant analysis” as a method for split point selection. It is an unbiased variable selection method described in Loh and Shih (1997). However, in advanced mode, the user can select “exhaustive search” (Breiman et al., 1984) which is used in CART. The former is the default option if the number of classes is more than 2, otherwise the latter is the default option. If the latter option is selected, the program will ask for the user to choose the splitting criterion including one of the following five methods which are studied in Shih (1999):

```
1 Likelihood Ratio G^2
2 Pearson Chi^2
3 Gini
4 MPI (Mean Posterior Improvement)
5 Other members of the divergence family
```

The likelihood criterion is the default option. If instead the CART-style split is used, the Gini criterion is the default option. In our case, we selected the fifth method of exhaustive search which was optimal regarding the misclassification ratio.

QUEST asks for the prior for each class. If the priors are to given, the program will then ask the user to input the priors. If unequal costs are present (like in this example), the priors are altered using the formula in Breiman et al. (1984, pp. 114-115). In our cases the prior for each class is estimated based on the class distribution. This asks for the misclassification costs. If the costs are to be given, the program will ask the user to input the costs. In our cases the misclassification costs are equal. The user can choose either split on a single variable or linear combination of variables. We used split on a single variable.

QUEST also asks for the number of SEs which controls the size of the pruned tree. 0-SE gives the tree with the smallest cross-validation estimate of misclassification cost or error. QUEST enables user to select the value of V in V-fold cross-validation. The larger the value of V, the longer running time the program takes to run. 10-fold and 226-fold (Leave-One-Out) are used in our cases.

The classification matrices based on the learning sample and CV procedure are reported. Some samples of these reports are shown in Appendix A. You can see a table gives the sequence of pruned subtrees. The 3rd column shows the cost complexity value for each subtree by using the definition in Breiman et al. (1984, Definition 3.5 p. 66). The 4th column gives the current or re-substitution cost (error) for each subtree. Another table gives the size, estimate of misclassification cost and its standard error for each pruned sub-tree. The 2nd column shows the number of terminal nodes. The 3rd column shows the mean cross-validation estimate of misclassification cost and the 4th column gives its estimated standard error using the approximate formula in Breiman et al. (1984, pp. 306-309). The tree marked with an "*" is the one with the minimum mean cross-validation estimate of misclassification cost (also called the 0-SE tree). The tree based on the mean cross-validation estimate of misclassification cost and the number of SEs is marked with "**" (See Appendix A).

QUEST trees are given in outline form suitable for importing into flowchart packages like *allCLEAR* (CLEAR Software, 1996). Alternatively, the trees may be outputted in LaTeX code. The public domain macro package pstricks (Goossens, Rahtz and Mittelbach, 1997) or TreeTEX (Bruggemann-Klein and Wood, 1988) is needed to render the LaTeX trees.

CRUISE is also a new statistical decision tree algorithm for classification and data mining. It has negligible bias in variable selection. It splits each node into as many sub-nodes as the number of classes in the response variable It has several ways to deal with missing values. It can detect local interactions between pairs of predictor variables.

CRUISE has most of QUEST capabilities and reports (See Appendix A). We have brought the results of tree-based classification with QUEST and the CRUISE into the final reporting table (Table 4.11), which includes all tree-based and non-tree based classifiers on our data set in the cases of 2-Classes, 3-Classes, and 9-Classes.

4.2.3 Final Results without optimization

The overall results of classifiers' performance on our data set are shown in the Table 4.11. Regarding individual classifier, for the case of 2-classes, *k*NN has the best performance with **82.3%** accuracy. In the case of 3-classes and 9-classes, CART has the best accuracy of about **60%** in 3-classes and **43%** in 9-Classes. However, considering the combination of non-tree-based classifiers, the CMC has the best performance in all three cases. That is we got the **86.8%** accuracy in the case of 2-Classes, **71%** in the case of 3-Classes, and **51%** in the case of 9-Classes.

Table 4.11: Comparing the Error Rate of all classifiers on PHY183 data set in the cases of 2-Classes, 3-Classes, and 9-Classes, using 10-fold cross-validation, *Without Optimization*

		Error Rate		
Classifier		2-Classes	3-Classes	9-Classes
Tree Classifier	C5.0	20.7%	43.2%	74.4%
	CART	18.5%	40.1%	66.9%
	QUEST	19.5%	42.9%	80.0%
	CRUISE	19.0%	45.1%	77.1%
Non-tree Classifier	Bayes	23.6%	51.4%	77.0%
	1NN	23.2%	49.5%	71.0%
	kNN	17.7%	49.6%	72.5%
	Parzen	25.0%	51.9%	79.5%
	MLP	20.5%	49.1%	-
	CMC	13.2%	29.1%	49.0%

So far, we have grouped students using multiple classifiers, comparing their prediction accuracy with CMC. In the next section we will study a way to optimize these results in order to find more efficient and more accurate classifiers.

4.3 Optimizing the prediction accuracy

We found that a combination of multiple classifiers leads to a significant improvement in classification performance. Through weighting the feature vectors using a Genetic Algorithm we can optimize the prediction accuracy and get a marked improvement over raw classification. We further show that when the number of features is few; feature-weighting works better than just feature selection.

4.3.1 Genetic Algorithms (GAs)

This learning procedure can be considered a search through a space of data points. A genetic algorithm presents a powerful alternative to traditional search techniques. It has been inspired by a similar predictive model in nature. Evolution in nature is controlled through the following principles:

Natural Selection: the strongest specimens have the highest chance to survive and reproduce, while the weak ones are likely to die before the reproduction stage.

Reproduction: the fittest specimens recombine their genetic information, thus creating new specimens with somewhat new characteristics.

Mutation leads to random changes in genetic information.

The success of this “search technique” in nature inspired some researchers to propose methods and develop algorithms that could be encoded in computer programs. A clear and simple introduction to the discipline of genetic algorithm has been made by Goldberg (1989). To start casting a real problem in a setting where its solution is can be obtained through a genetic algorithm, two important steps are taken: encoding the search space into chromosomes (a string of binary/real values); and defining a *fitness function* that plays the role of an evaluation function in a heuristic search. The implementation of a chromosome is typically in the form of bit strings.

4.3.1.1 What is a Simple GA (SGA)?

The main steps of a GA are reproduction, recombination, and mutation in the following algorithm (Michalski et al., 1998):

1. Construct the initial population as a set of binary strings generated randomly or by some pre-specified mechanisms.
2. Replicate the specimen in the population into a set of survivors by a mechanism that ensures that specimens with a higher fitness value have a higher chance of survival.
3. Pair up all the survivors, such that each has a mate. Next specific chunks of encoded data are swapped between mates. Mutation arises when a single bit flip-flops.
4. If the fitness function has not improved through several cycles, stop; otherwise go to step 2.

4.3.1.2 Specific use of GAs in pattern classification

Genetic Algorithms have been shown to be an effective tool to use in data mining and pattern recognition (Freitas, 2002; Jain and Zongker, 1997; Falkenauer, 1998; Pei et al., 1997; Park and Song, 1998; Michalewicz, 1996; De Jong et al., 1993). An important aspect of GAs in a learning context is their use in pattern recognition. There are two different approaches to applying GA in pattern recognition:

1. Apply a GA directly as a classifier. Bandyopadhyay and Murthy (1995) applied GA to find the decision boundary in N dimensional feature space.
2. Use a GA as an optimization tool for resetting the parameters in other classifiers. Most applications of GAs in pattern recognition optimize some parameters in the classification process.

Many researchers have used GAs in feature selection (Bala et al. 1997; Guerra-Salcedo and Whitley 1999, Vafaie and De Jong, 1993; Martin-Bautista and Vila, 1999). GAs have been applied to find an optimal set of feature weights that improve classification accuracy. First, a traditional feature extraction like Principal Component Analysis (PCA) is applied, and then a classifier like *k*-NN is used to calculate the fitness function for GA (Seidlecki, 1989; Pei et al., 1998). Combined classifiers are another area that GAs have been used to optimize. Kuncheva and Jain (2000) used a GA for selecting the features as well as selecting the types of individual classifiers in their design of a Classifier Fusion System. GA is also used in selecting the prototypes in the case-based classification (Skalak, 1994).

In this work we focus on the second approach and use a GA to optimize a combination of classifiers. Our objective is to *predict* the students' final grades based on their web-use features, which are extracted from the homework data. We design, implement, and evaluate a series of pattern classifiers with various parameters in order to compare their performance on a data set from LON-CAPA. Error rates for the individual classifiers, their combination and the GA optimized combination are presented.

4.3.2 Implementation of a GA to optimize the prediction accuracy

We use the GAToolBox[12] from MATLAB to implement a GA to optimize classification performance. Our goal is to find a population of best weights for every feature vector, which minimize the classification error rate.

The feature vector for our predictors are the set of six variables for every student: Success rate, Success at the first try, Number of attempts before correct answer is derived, the time at which the student got the problem correct relative to the due date, total time spent on the problem, and the number of online interactions of the student both with other students and with the instructor.

We randomly initialize a population of six dimensional weight vectors with values between 0 and 1, corresponding to the feature vector and experimented with different number of population sizes. We obtained good results using a population with 200 individuals. The GA Toolbox supports binary, integer, real-valued and floating-point chromosome representations. Real-valued populations may be initialized using the toolbox function *crtrp*. For example, to create a random population of 6 individuals with 200 variables each: we define boundaries on the variables in *FieldD* which is a matrix containing the boundaries of each variable of an individual.

```
FieldD = [ 0 0 0 0 0 0;  % lower bound
           1 1 1 1 1 1]; % upper bound
```

An initial population created with `Chrom = crtrp(200, FieldD)`, An example is as follows:

[12] Downloaded from http://www.shef.ac.uk/~gaipp/ga-toolbox/

```
Chrom = 0.23 0.17 0.95 0.38 0.06 0.26
        0.35 0.09 0.43 0.64 0.20 0.54
        0.50 0.10 0.09 0.65 0.68 0.46
        0.21 0.29 0.89 0.48 0.63 0.89
.................
```

We use the simple genetic algorithm (SGA), which is described by Goldberg (1989).

4.3.2.1 GA Operators

The SGA uses common GA operators to find a population of solutions which optimize the fitness values.

4.3.2.1.1 Recombination

We use "*Stochastic Universal Sampling*" (Baker, 1987) as our selection method. A form of stochastic universal sampling is implemented by obtaining a cumulative sum of the fitness vector, *FitnV*, and generating *N* equally spaced numbers between 0 and sum(FitnV). Thus, only one random number is generated, all the others used being equally spaced from that point. The index of the individuals selected is determined by comparing the generated numbers with the cumulative sum vector. The probability of an individual being selected is then given by

$$F(x_i) = \frac{f(x_i)}{\sum_{i=1}^{N_{ind}} f(x_i)}$$

where $f(x_i)$ is the fitness of individual x_i and $F(x_i)$ is the probability of that individual being selected.

4.3.2.1.2 Crossover

The crossover operation is not necessarily performed on all strings in the population. Instead, it is applied with a probability *Px* when the pairs are chosen for breeding. We select *Px* = 0.7. There are several functions to make crossover on real-valued matrices. One of them is *recint*, which performs intermediate recombination between pairs of individuals in the current population, OldChrom, and returns a new population after mating, NewChrom. Each row of OldChrom corresponds to one individual. *recint* is a function only applicable to populations of real-value variables. Intermediate recombination combines parent values using the following formula (Muhlenbein and Schlierkamp-Voosen, 1993).

Offspring = parent1 + Alpha × (parent2 – parent1)

Alpha is a Scaling factor chosen uniformly in the interval [-0.25, 1.25]

4.3.2.1.3 Mutation

A further genetic operator, mutation is applied to the new chromosomes, with a set probability *Pm*. Mutation causes the individual genetic representation to be changed according to some probabilistic rule. Mutation is generally considered to be a background operator that ensures that the probability of searching a particular subspace of the problem space is never zero. This has the effect of tending to inhibit the possibility of converging to a local optimum, rather than the global optimum.

There are several functions to make mutation on real-valued population. We used *mutbga,* which takes the real-valued population, OldChrom, mutates each variable with given probability and returns the population after mutation, *NewChrom* = *mutbga(OldChrom, FieldD, MutOpt)* takes the current population, stored in the matrix

OldChrom and mutates each variable with probability by addition of small random values (size of the mutation step). We considered 1/600 as our mutation rate. The mutation of each variable is calculated as follows:

Mutated Var = Var + MutMx × range × MutOpt(2) × delta

where delta is an internal matrix which specifies the normalized mutation step size; MutMx is an internal mask table; and MutOpt specifies the mutation rate and its shrinkage during the run. The mutation operator *mutbga* is able to generate most points in the hypercube defined by the variables of the individual and the range of the mutation. However, it tests more often near the variable, that is, the probability of small step sizes is greater than that of larger step sizes.

4.3.2.2 Fitness Function

During the reproduction phase, each individual is assigned a fitness value derived from its raw performance measure given by the objective function. This value is used in the selection to bias towards more fit individuals. Highly fit individuals, relative to the whole population, have a high probability of being selected for mating whereas less fit individuals have a correspondingly low probability of being selected. The error rate is measured in each round of cross validation by dividing “the total number of misclassified examples” into “total number of test examples”. Therefore, our *fitness function* measures the error rate achieved by CMC and our objective would be to maximize this performance (minimize the error rate).

4.3.3 Experimental Results of GA Optimization

For GA optimization, we used 200 individuals in our population, running the GA over 500 generations. We ran the program 10 times and got the averages, which are shown, in Table 4.12. In every run 500×200 times the fitness function is called in which we used 10-fold cross validation to measure the average performance of CMC. So every classifier is called 3×10^6 times for the case of 2-classes, 3-classes and 9-classes. Thus, the time overhead for fitness evaluation is critical. Since using the MLP in this process took about 2 minutes and all other four non-tree classifiers (Bayes, 1NN, 3NN, and Parzen window) took only 3 seconds, we omitted the MLP from our classifiers group so we could obtain the results in a reasonable time.

Figures 4.6-4.8 shows the best result of the ten runs over our data set. These figures represent the population mean, the best individual at each generation and the best value yielded by the run. The results in Table 4.12 represent the mean performance with a two-tailed t-test with a 95% confidence interval. For the improvement of GA over non-GA result, a P-value indicating the probability of the Null-Hypothesis (There is no improvement) is also given, showing the significance of the GA optimization.

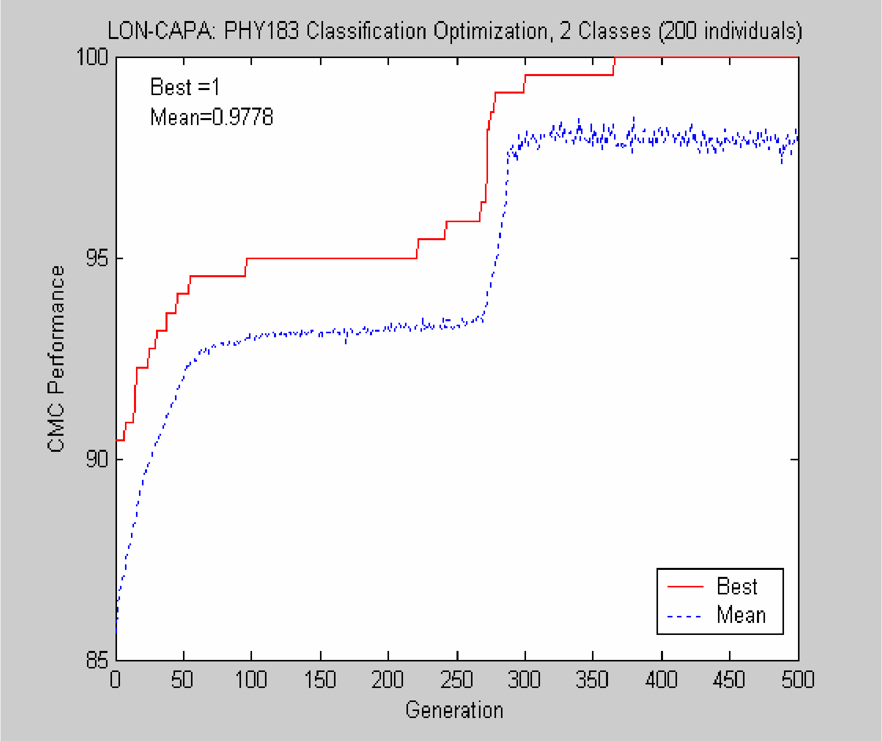


Figure 4.6. Graph of GA Optimized CMC performance in the case of 2-Classes

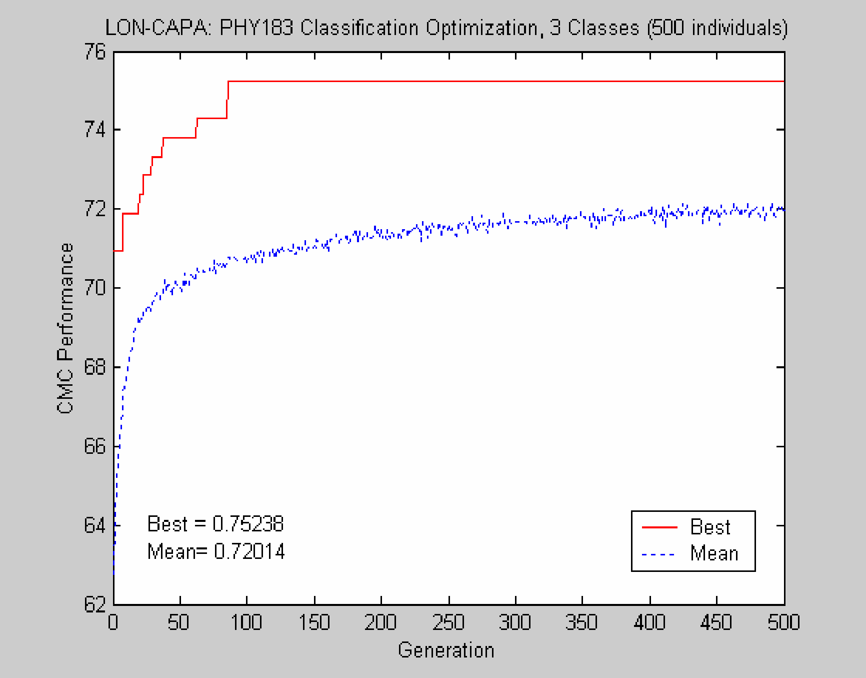


Figure 4.7. Graph of GA Optimized CMC performance in the case of 3-Classes

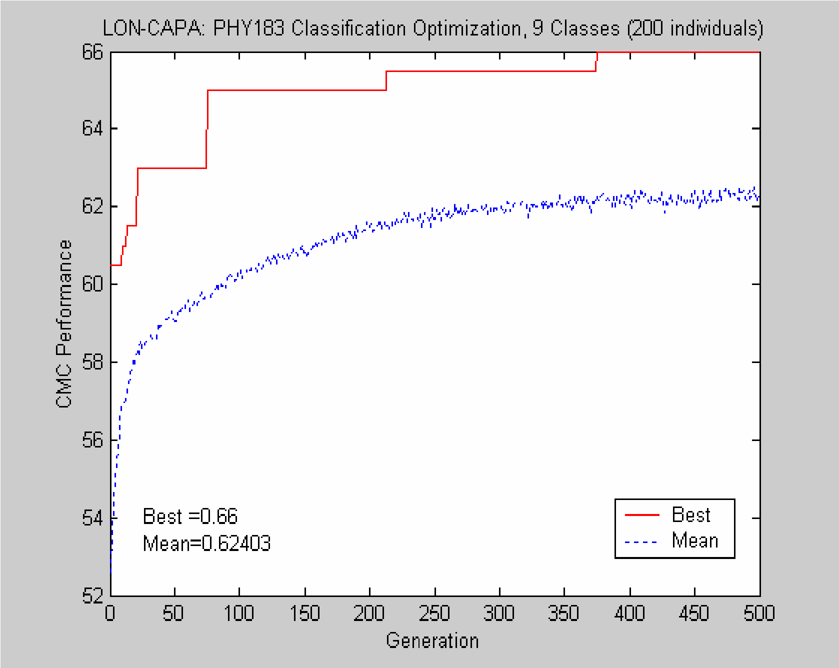


Figure 4.8. Graph of GA Optimized CMC performance in the case of 9-Classes

Table 4.12. Comparing the CMC Performance on PHY183 data set Using GA and without GA in the cases of 2-Classes, 3-Classess, and 9-Classes, 95% confidence interval.

	Performance %		
Classifier	2-Classes	3-Classes	9-Classes
CMC of four Classifiers without GA	83.87 ± 1.73	61.86 ± 2.16	49.74 ± 1.86
GA Optimized CMC, Mean individual	94.09 ± 2.84	72.13 ± 0.39	62.25 ± 0.63
Improvement	10.22 ± 1.92	10.26 ± 1.84	12.51 ± 1.75

All have p<0.000, indicating significant improvement. Therefore, using GA, in all the cases, we got more than a 10% mean individual performance improvement and about 12 to 15% mean individual performance improvement. Figure 4.9 shows the graph of average mean individual performance improvement.

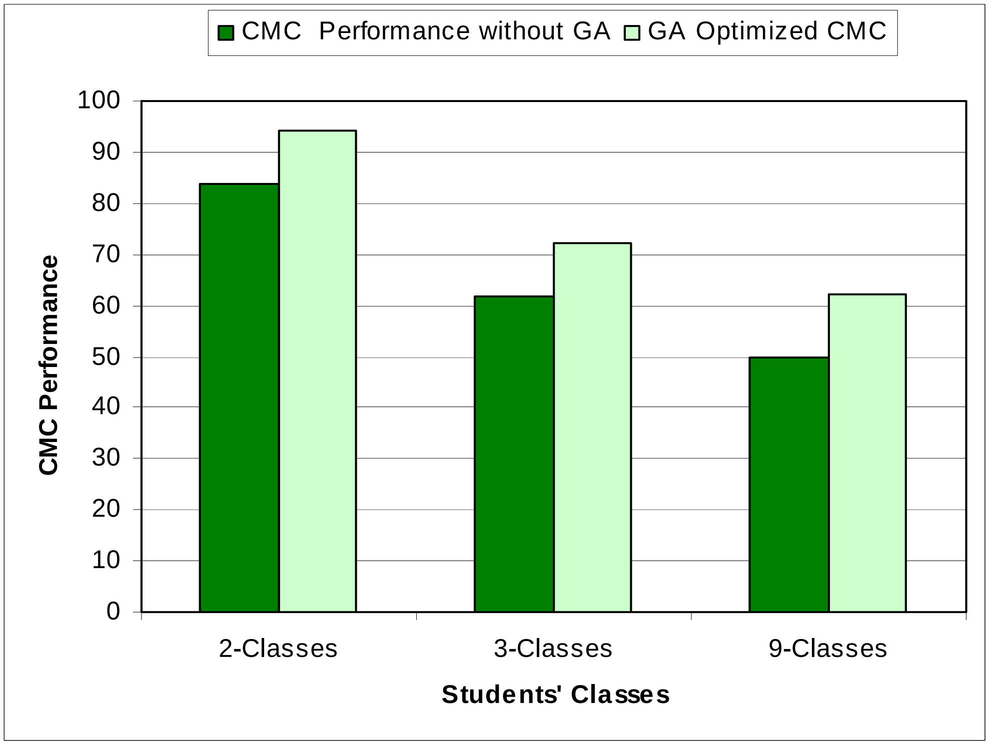


Figure 4.9. Char t of comparing CMC average performance, using GA and without GA.

Finally, we can examine the individuals (weights) for features by which we obtained the improved results. This feature weighting indicates the importance of each feature for making the required classification. In most cases the results are similar to Multiple Linear Regressions or tree-based software that use statistical methods to measure feature importance. Table 4.13 shows the importance of the six features in the 3-classes case using the Entropy splitting criterion. Based on entropy, a statistical property called *information gain* measures how well a given feature separates the training examples in relation to their target classes. Entropy characterizes *impurity* of an arbitrary collection of examples S at a specific node N. In Duda et al. (2001) the impurity of a node *N* is denoted by *i(N)*.

$$Entropy(S) = i(N) = -\sum_j P(\omega_j)\log_2 P(\omega_j)$$

where $P(\omega_j)$ is the fraction of examples at node N that go to category ω_j.

Table 4.13. Feature Importance in 3-Classes Using Entropy Criterion

Feature	Importance %
Total_Correct _Answers	100.00
Total_Number_of_Submissions	58.61
First_Got_Correct	27.70
Time_Spent_to_Solve	24.60
Total_Time_Spent	24.47
Communication	9.21

The GA results also show that the “Total number of correct answers” and the “Total number of submissions” are the most important features for classification accuracy; both are positively correlated to the true class labels. The second column in Table 4.13 shows the percentage of feature importance. One important finding is that GAs determine optimal weights for features. Using this set of weights we can extract a new set of features which significantly improve the prediction accuracy. In other words, this resultant set of weights transforms the original features into new salient features.

4.4 Extending the work toward more LON-CAPA data sets

We selected 14 student/course data sets of MSU courses, which used LON-CAPA as shown in Table 4.14 and 4.15.

Table 4.14. 14 of LON-CAPA courses at MSU

Course	Term	Title
ADV 205	SS03	Principles of Advertising
BS 111	SS02	Biological Science: Cells and Molecules
BS 111	SS03	Biological Science: Cells and Molecules
CE 280	SS03	Civil Engineering: Intro Environment Eng.
FI 414	SS03	Advanced Business Finance (w)
LBS 271	FS02	Lyman Briggs School: Physics I
LBS 272	SS03	Lyman Briggs School: Physics II
MT 204	SS03	Medical Tech.: Mechanisms of Disease
MT 432	SS03	Clinic Immun. & Immunohematology
PHY 183	SS02	Physics Scientists & Engineers I
PHY 183	SS03	Physics Scientists & Engineers I
PHY 231c	SS03	Introductory Physics I
PHY 232c	FS03	Introductory Physics II
PHY 232	FS03	Introductory Physics II

Table 4.15 Characteristics of 14 of MSU courses, which held by LON-CAPA

Course	Number of Students	Number of Problems	Size of Activity log	Size of useful data	Number of Transactions
ADV205_SS03	609	773	82.5 MB	12.1 MB	424,481
BS111_SS02	372	229	361.2 MB	34.1 MB	1,112,394
BS111_SS03	402	229	367.6 MB	50.2 MB	1,689,656
CE280_SS03	178	19 6	28.9 MB	3.5 MB	127,779
FI414_SS03	169	68	16.8 MB	2.2 MB	83,715
LBS271_FS02	132	174	119.8 MB	18.7 MB	706,700
LBS272_SS03	102	166	73.9 MB	15.3 MB	585,524
MT204_SS03	27	150	5.2 MB	0.7 MB	23,741
MT432_SS03	62	150	20.0 MB	2.4 MB	90,120
PHY183_SS02	227	184	140.3 MB	21.3 MB	452,342
PHY183_SS03	306	255	210.1 MB	26.8 MB	889,775
PHY231c_SS03	99	247	67.2 MB	14.1 MB	536,691
PHY232c_SS03	83	194	55.1 MB	10.9 MB	412,646
PHY232_FS03	220	259	138.5 MB	19.7 MB	981,568

For example, the third row of the Table 4.16 shows that BS111 (Biological Science: Cells and Molecules) was held in spring semester 2003 and contained 229 online homework problems, and 402 students used LON-CAPA for this course. The BS111 course had an activity log with approximately 368 MB. Using some Perl script modules for cleansing the data, we found 48 MB of useful data in the BS111 SS03 course. We then pulled from these logged data 1,689,656 transactions (interactions between students and homework/exam/quiz problems) from which we extracted the following nine features (Having revised six features that were explained in 4.1):

1. Number of attempts before correct answer is derived
2. Total number of correct answers
3. Success at the first try
4. Getting the problem correct on the second try
5. Getting the problem correct between 3 and 9 tries

6. Getting the problem correct with a high number of tries (10 or more tries).
7. Total time that passed from the first attempt, until the correct solution was demonstrated, regardless of the time spent logged in to the system
8. Total time spent on the problem regardless of whether they got the correct answer or not
9. Participating in the communication mechanisms

Based on the above extracted features in each course, we classify the students, and try to predict for every student to which class he/she belongs. We categorize the students with one of two class labels: "*Passed*" for grades higher than 2.0, and "*Failed*" for grades less than or equal to 2.0 where the MSU grading system is based on grades from 0.0 to 4.0. Figure 4.10 shows the grade distribution for the BS111 fall semester 2003.

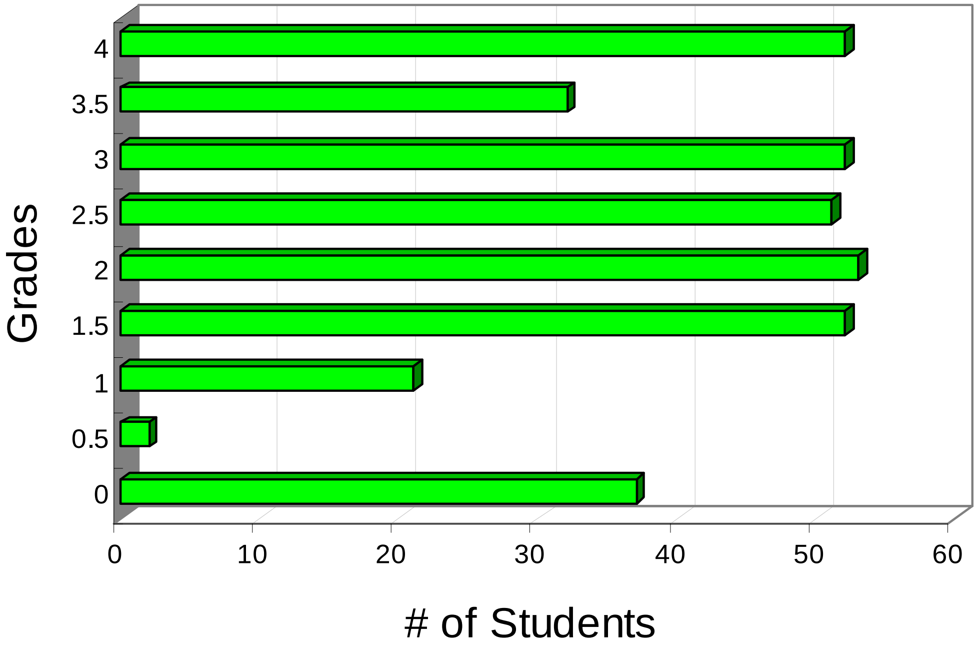


Figure 4.10. LON-CAPA: BS111 SS03, Grades distribution

4.4.1 Experimental Results

Without using GA, the overall results of classification performance on our datasets for four classifiers and classification fusion are shown in the Table 4.16. Individual classifiers, 1NN and *k*NN have mostly the best performance. However, the classification fusion improved the classification accuracy significantly in all data sets. That is, it achieved in average 79% accuracy over the given data sets.

Table 4.16 Comparing the average performance% of ten runs of classifiers on the given datasets using 10-fold cross validation, without GA

Data sets	Bayes	1NN	*k*NN	Parzen Window	Classification **Fusion**
ADV 205, 03	55.7	69.9	70.7	55.8	78.2
BS 111, 02	54.6	67.8	69.6	57.6	74.9
BS 111, 03	52.6	62.1	55.0	59.7	71.2
CE 280, 03	66.6	73.6	74.9	65.2	81.4
FI 414, 03	65.0	76.4	72.3	70.3	82.2
LBS 271, 02	66.9	75.6	73.8	59.6	79.2
LBS 272, 03	72.3	70.4	69.6	65.3	77.6
MT 204, 03	63.4	71.5	68.4	56.4	82.2
MT 432, 03	67.6	77.6	79.1	59.8	84.0
PHY 183, 02	73.4	76.8	80.3	65.0	83.9
PHY 183, 03	59.6	66.5	70.4	54.4	76.6
PHY 231c, 03	56.7	74.5	72.6	60.9	80.7
PHY 232c, 03	65.6	71.7	75.6	57.8	81.6
PHY 232, 03	59.9	73.5	71.4	56.3	79.8

For GA optimization, we used 200 individuals (weight vectors) in our population, running the GA over 500 generations. We ran the program 10 times and got the averages, which are shown, in Table 4.17.

Table 4.17 Comparing the classification fusion performance on given datasets, without-GA, using-GA (Mean individual) and improvement, 95% confidence interval

Data sets	Without GA	GA optimized	Improvement
ADV 205, 03	78.19 ± 1.34	89.11 ± 1.23	10.92 ± 0.94
BS 111, 02	74.93 ± 2.12	87.25 ± 0.93	12.21 ± 1.65
BS 111, 03	71.19 ± 1.34	81.09 ± 2.42	9.82 ± 1.33
CE 280, 03	81.43 ± 2.13	92.61 ± 2.07	11.36 ± 1.41
FI 414, 03	82.24 ± 1.54	91.73 ± 1.21	9.50 ± 1.76
LBS 271, 02	79.23 ± 1.92	90.02 ± 1.65	10.88 ± 0.64
LBS 272, 03	77.56 ± 0.87	87.61 ± 1.03	10.11 ± 0.62
MT 204, 03	82.24 ± 1.65	91.93 ± 2.23	9.96 ± 1.32
MT 432, 03	84.03 ± 2.13	95.21 ± 1.22	11.16 ± 1.28
PHY 183, 02	83.87 ± 1.73	94.09 ± 2.84	10.22 ± 1.92
PHY 183, 03	76.56 ± 1.37	87.14 ± 1.69	9.36 ± 1.14
PHY 231c, 03	80.67 ± 1.32	91.41 ± 2.27	10.74 ± 1.34
PHY 232c, 03	81.55 ± 0.13	92.39 ± 1.58	10.78 ± 1.53
PHY 232, 03	79.77 ± 1.64	88.61 ± 2.45	9.13 ± 2.23
Total Average	78.98 ± 12	90.03 ± 1.30	10.53 ± 56

The results in Table 4.17 represent the mean performance with a two-tailed t-test with a 95% confidence interval for every data set. For the improvement of GA over non-GA result, a P-value indicating the probability of the Null-Hypothesis (There is no improvement) is also given, showing the significance of the GA optimization. All have p<0.000, indicating significant improvement. Therefore, using GA, in all the cases, we got approximately more than a 10% mean individual performance improvement and about 10 to 17% best individual performance improvement. Fig. 4.11 shows the results of one of the ten runs in the case of 2-Classes (passed and failed). The doted line represents the population mean, and the solid line shows the best individual at each generation and the best value yielded by the run.

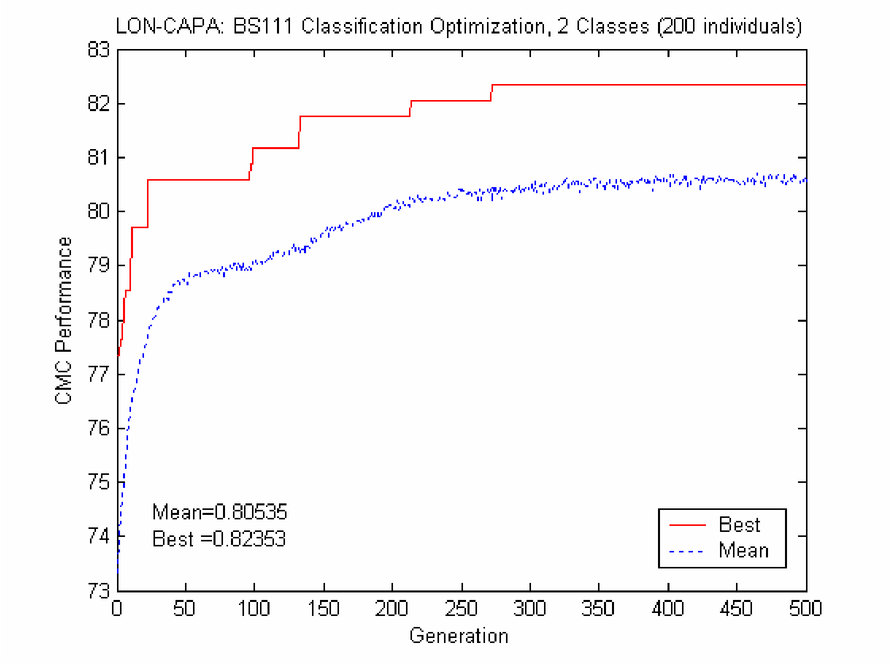


Figure 4.11 GA-Optimized Combination of Multiple Classifiers' (CMC) performance in the case of 2-Class labels (Passed and Failed) for BS111 2003, 200 weight vectors individuals, 500 Generations

Finally, we can examine the individuals (weights) for features by which we obtained the improved results. This feature weighting indicates the *importance* of each feature for making the required classification. In most cases the results are similar to Multiple Linear Regressions or some tree-based software (like CART) that use statistical methods to measure feature importance. The GA feature weighting results, as shown in Table 4.18, state that the "Success with high number of tries" is the most important feature. The "Total number of correct answers" feature is also the most important in some cases; both are positively correlated to the true class labels.

If we use one course as the training data and another course as the test data we again achieve a significant improvement in prediction accuracy for both using the combination of multiple classifiers and applying genetic algorithms as the optimizer. For example, using BS111 from fall semester 2003 as the training set and PHY231 from spring semester 2004 as the test data, and using the weighted features in the training set, we obtain a significant improvement for classification accuracy in the test data.

Table 4.18 Relative Feature Importance%, Using GA weighting for BS111 2003 course

Feature	Importance %
Average Number of Tries	18.9
Total number of Correct Answers	84.7
# of Success at the First Try	24.4
# of Success at the Second Try	26.5
Got Correct with 3-9 Tries	21.2
Got Correct with # of Tries ≥ 10	91.7
Time Spent to Solve the Problems	32.1
Total Time Spent on the Problems	36.5
# of communication	3.6

Table 4.19 shows the importance of the nine features in the BS 111 SS03 course, applying the Gini splitting criterion. Based on Gini, a statistical property called information gain measures how well a given feature separates the training examples in relation to their target classes. Gini characterizes impurity of an arbitrary collection of examples S at a specific node N. In Duda et al. (2001) the impurity of a node N is denoted by *i*(N) such that:

$$\text{Gini(S)} = i(N) = \sum_{j \neq i} P(\omega_j)P(\omega_i) = 1 - \sum_j P^2(\omega_j)$$

where $P(\omega_j)$ is the fraction of examples at node *N* that go to category ω_j. Gini attempts to separate classes by focusing on one class at a time. It will always favor working on the largest or, if you use costs or weights, the most important class in a node.

Table 4.19 Feature Importance for BS111 2003, using decision-tree software CART, applying Gini Criterion

Variable		
Total number of Correct Answers	100.00	||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Got Correct with # of Tries ≥ 10	93.34	||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Average number of tries	58.61	|||||||||||||||||||||||||||||||||||
# of Success at the First Try	37.70	||||||||||||||||||
Got Correct with 3-9 Tries	30.31	||||||||||||||
# of Success at the Second Try	23.17	||||||||
Time Spent to Solve the Problems	16.60	|||||
Total Time Spent on the Problems	14.47	||||
# of communication	2.21	|

Comparing results in Table 4.18 (GA-weighting) and Table 4.19 (Gini index criterion) shows the very similar output, which demonstrates merits of the proposed method for detecting the feature importance.

4.5 Summary

We proposed a new approach to classifying student usage of web-based instruction. Four classifiers were used to segregate student data. A combination of multiple classifiers led to a significant accuracy improvement in all three cases (2-, 3- and 9-Classes). Weighting the features and using a genetic algorithm to minimize the error rate improved the prediction accuracy by at least 10% in the all cases. In cases where the number of features was low, feature weighting was a significant improvement over selection. The successful optimization of student classification in all three cases demonstrates the value of LON-CAPA data in predicting students' final grades based on features extracted from homework data. This approach is easily adaptable to different types of courses, different population sizes, and allows for different features to be analyzed. This work represents a rigorous application of known classifiers as a means of analyzing and comparing use and performance of students who have taken a technical course that was partially/completely administered via the web.

For future work, we plan to implement such an optimized assessment tool for every student on any particular problem. Therefore, we can track students' behaviors on a particular problem over several semesters in order to achieve more reliable prediction. This work has been published in (Minaei-Bidgoli & Punch, 2003; Minaei-Bidgoli, et al. 2003; Minaei-Bidgoli et al., 2004c-e).

Chapter 5 Ensembles of Multiple Clusterings

Since LON-CAPA data are distributed among several servers and distributed data mining requires efficient algorithms form multiple sources and features, this chapter represents a framework for clustering ensembles in order to provide an optimal framework for categorizing distributed web-based educational resources. This research extends previous theoretical work regarding clustering ensembles with the goal of creating an optimal framework for categorizing web-based educational resources. Clustering ensembles combine multiple partitions of data into a single clustering solution of better quality. Inspired by the success of supervised bagging and boosting algorithms, we propose non-adaptive and adaptive resampling schemes for the integration of multiple independent and dependent clusterings. We investigate the effectiveness of bagging techniques, comparing the efficacy of sampling with and without replacement, in conjunction with several consensus algorithms. In our adaptive approach, individual partitions in the ensemble are sequentially generated by clustering specially selected subsamples of the given data set. The sampling probability for each data point dynamically depends on the consistency of its previous assignments in the ensemble. New subsamples are then drawn to increasingly focus on the problematic regions of the input feature space. A measure of data point clustering consistency is therefore defined to guide this adaptation. Experimental results show improved stability and accuracy for clustering structures obtained via bootstrapping, subsampling, and adaptive techniques. A meaningful consensus partition for an entire set of data points emerges from multiple clusterings of bootstraps and subsamples. Subsamples of small size can reduce computational cost and measurement complexity for many unsupervised data mining tasks with distributed sources of data. This empirical study also compares the performance of adaptive and non-adaptive clustering ensembles using different consensus functions on a number of data sets. By focusing attention on the data points with the least consistent clustering assignments, one can better approximate the inter-cluster boundaries and improve clustering accuracy and convergence speed as a function of the number of partitions in the ensemble. The comparison of adaptive and non-adaptive approaches is a new avenue for research, and this study helps to pave the way for the useful application of distributed data mining methods.

5.1 Introduction

Exploratory data analysis and, in particular, data clustering can significantly benefit from combining multiple data partitions. Clustering ensembles can offer better solutions in terms of robustness, novelty and stability (Fred & Jain, 2002; Strehl & Ghosh, 2002; Topchy et al., 2003a). Moreover, their parallelization capabilities are a natural fit for the demands of distributed data mining. Yet, achieving stability in the combination of multiple clusterings presents difficulties.

The combination of clusterings is a more challenging task than the combination of supervised classifications. In the absence of labeled training data, we face a difficult correspondence problem between cluster labels in different partitions of an ensemble.

Recent studies (Topchy et al., 2004) have demonstrated that consensus clustering can be found outside of voting-type situations using graph-based, statistical or information-theoretic methods without explicitly solving the label correspondence problem. Other empirical consensus functions were also considered in (Dudoit & Fridlyand, 2003; Fisher & Buhmann, 2003, Fern & Brodley, 2003). However, the problem of consensus clustering is known to be NP complete (Barthelemy & Leclerc, 1993).

Beside the consensus function, clustering ensembles need a partition generation procedure. Several methods are known to create partitions for clustering ensembles. For example, one can use:

1. different clustering algorithms (Strehl & Ghosh, 2002),
2. different initializations – parameter values or built-in randomness of a specific clustering algorithm (Fred & Jain, 2002)
3. different subsets of features (weak clustering algorithms) (Topchy et al., 2003),
4. different subsets of the original data (data resampling) (Dudoit & Fridlyand, 2003; Fisher & Buhmann, 2003, Minaei et al., 2003).

The focus of this study is the last method, namely the combination of clusterings using random samples of the original data. Conventional data resampling generates ensemble partitions independently; the probability of obtaining the ensemble consisting of B partitions $\{\pi_1, \pi_2,\ldots,\pi_B\}$ of the given data, D, can be factorized as:

$$p(\{\pi_1, \pi_2, \ldots, \pi_B\} \mid D) = \prod_{t=1}^{B} p(\pi_t \mid D) \qquad (5.1)$$

.

Hence, the increased efficacy of an ensemble is mostly attributed to the number of independent, yet identically distributed partitions, assuming that a partition of data is treated as a random variable π. Even when the clusterings are generated sequentially, it is traditionally done without considering previously produced clusterings:

$$p(\pi_t \mid \pi_{t-1}, \pi_{t-2}, \ldots, \pi_1; D) = p(\pi_t \mid D) \tag{5.2}$$

However, similar to the ensembles of supervised classifiers using boosting algorithms (Brieman 1998), a more accurate consensus clustering can be obtained if contributing partitions take into account the previously determined solutions. Unfortunately, it is not possible to mechanically apply the decision fusion algorithms from the supervised (classification) to the unsupervised (clustering) domain. New objective functions for guiding partition generation and the subsequent decision integration process are necessary in order to guide further refinement. Frossyniotis et al. (2004) apply the general principle of boosting to provide a consistent partitioning of a data set. At each boosting iteration, a new training set is created and the final clustering solution is produced by aggregating the multiple clustering results through a weighted voting.

We propose a simple adaptive approach to partition generation that makes use of clustering history. In clustering, ground truth in the form of class labels is not available. Therefore, we need an alternative measure of performance for an ensemble of partitions. We determine clustering consistency for data points by evaluating a history of cluster assignments for each data point within the generated sequence of partitions. Clustering consistency serves for adapting the data sampling to the current state of an ensemble during partition generation. The goal of adaptation is to improve confidence in cluster assignments by concentrating sampling distribution on problematic regions of the feature space. In other words, by focusing attention on the data points with the least consistent clustering assignments, one can better approximate (indirectly) the inter-cluster boundaries.

The main objectives of this chapter are four-fold:

1. to present a detailed taxonomy of clustering ensemble approaches (section 5.2),
2. to expose critical and unaddressed issues in applying resampling methods (section 5.5),
3. to provide a detailed comparison of bootstrap versus subsampling ensemble generation (section 5.7),
4. and finally to study adaptive partitioning ensembles (section 5.6).

The remainder of the chapter is devoted to different consensus functions used in our experiments (section 5.4), algorithms for resampling schemes (sections 5.3 and 5.6), addressing the problems of estimation of clustering consistency and finding a consensus clustering (section 5.6). Finally, we evaluate the performance of adaptive clustering ensembles (Section 5.8) on a number of real-world and artificial data sets in comparison with non-adaptive clustering ensembles of bootstrap partitions (Dudoit & Fridlyand, 2003; Fisher & Buhmann, 2003, Minaei-Bidgoli et al., 2003b).

5.2 Taxonomy of different approaches

A growing number of techniques have been applied to clustering combinations. A co-association consensus function was introduced for finding a combined partition in (Fred & Jain, 2002). The authors further studied combining *k*-means partitions with random initializations and a random number of clusters. Topchy et al. (2003) proposed new consensus functions related to intra-class variance criteria as well as the use of weak clustering components. Strehl and Ghosh (2002) have made a number of important contributions, such as their detailed study of hypergraph-based algorithms for finding consensus partitions as well as their object-distributed and feature-distributed formulations of the problem. They also examined the combination of partitions with a deterministic overlap of points between data subsets (non-random).

Resampling methods have been traditionally used to obtain more accurate estimates of data statistics. Efron (1979) generalized the concept of so-called "pseudo-samples" to sampling *with* replacement – the *bootstrap* method. Resampling methods such as bagging have been successfully applied in the context of supervised learning (Breiman 1996). Jain and Moreau (1987) employed bootstrapping in cluster analysis to estimate the number of clusters in a multi-dimensional data set as well as for evaluating cluster tendency/validity. A measure of consistency between two clusters is defined in (Levine & Moreau, 2001). Data resampling has been used as a tool for estimating the validity of clustering (Fisher & Buhmann, 2003; Dudoit & Fridlyand, 2001; Ben-Hur et al., 2002) and its reliability (Roth et al., 2002).

The taxonomy of different consensus functions for clustering combination is shown in Figure 5.2. Several methods are known to create partitions for clustering ensembles. This taxonomy presents solutions for the generative procedure as well. Details of the algorithms can be found in the listed references in Figure 5.1.

Generative mechanisms (How to obtain different components?)

1. Apply various clustering algorithms (Strehl & Ghosh, 2002)

2. Use a single algorithm

2.1. Different built-in initialization (Fred & Jain, 2002; Topchy et al. 2003b)

2.2. Different parameters (Fred & Jain, 2002)

2.3. Different subsets of data points

2.3.1. Deterministic subsets (Strehl & Ghosh, 2002)

2.3.2. Resampling (Dudoit & Fridlyand,2001;Monti et al. 2003; Fisher & Buhmann, 2003, Minaei-Bidgoli et al., 2003b)

2.3.2.1. Bootstrap (Sampling with replacement)

2.3.2.2. Subsampling (Sampling without replacement)

2.3.2.3. Adaptive scheme (Topchy et al., 2004; Frossyniotis et al., 2004)

2.4. Projecting data onto different subspaces (Topchy et al., 2003a; Zhang&Brodely, 2003)

2.5. Different subset of features (Strehl & Ghosh, 2002)

Consensus functions (How to integrate cluster ensemble?)

1. Using Co-association Matrix (Fred & Jain, 2002; Monti et al., 2003)

1.1. Single Link (SL)/ Minimum Spanning Tree (MST)

1.2. Complete Link (CL)

1.3. Average Link (AL)

1.4. Ward, or other similarity based algorithms

2. (Hyper) Graph Partitioning (Strehl & Ghosh, 2002)

2.1. Hyper Graph Partition Algorithm (HGPA)

2.2. Meta CLustering Algorithm (MCLA)

2.3. Clustering Similarity Partition Algorithm (CSPA)

3. Information-theoretic methods, e.g. Quadratic Mutual Information (Topchy et al. 2003a)

4. Voting Approach (Dudoit & Fridlyand,2001)

5. Mixture Model (Topchy et al. 2003b)

Figure 5.1 Different approaches to clustering combination

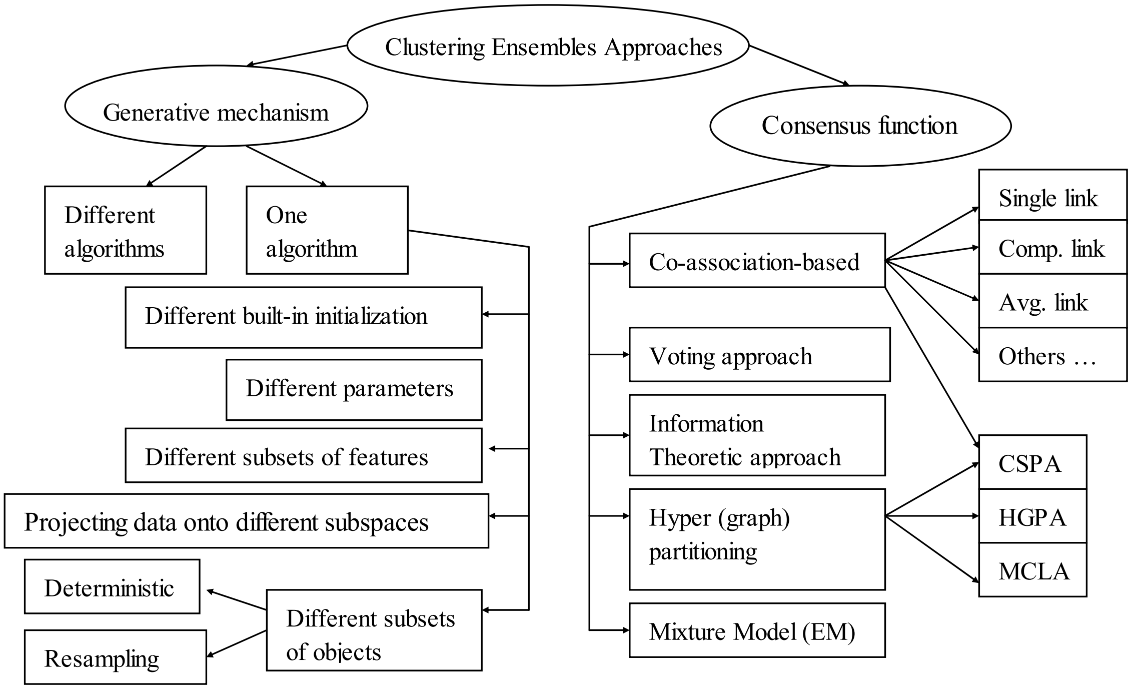

Clustering Ensembles Approaches
Generative mechanism
Consensus function
Different algorithms
One algorithm
Different built-in initialization
Different parameters
Different subsets of features
Projecting data onto different subspaces
Deterministic
Resampling
Different subsets of objects
Co-association-based
Voting approach
Information Theoretic approach
Hyper (graph) partitioning
Mixture Model (EM)
Single link
Comp. link
Avg. link
Others …
CSPA
HGPA
MCLA

Figure 5.2 Taxonomy of different approaches to clustering combination

It is a long-standing goal of clustering research to design scalable and efficient algorithms for large datasets (Zhang et al., 1996). One solution to the scaling problem is the parallelization of clustering by sharing processing among different processors (Zhang et al., 2000; Dhillon & Modha, 2000). Recent research in data mining has considered a fusion of the results from multiple sources of data or from data features obtained in a distributed environment (Park & Kargupta, 2003). Distributed data clustering deals with the combination of partitions from many data subsets (usually disjoint). The combined final clustering can be constructed centrally either by combining explicit cluster labels of data points or, implicitly, through the fusion of cluster prototypes (e.g., centroid-based). We analyze the first approach, namely, the clustering combination via consensus functions operating on multiple labelings of the different subsamples of a data set. This study seeks to answer the question of the optimal size and granularity of the component partitions.

5.3 Non-adaptive algorithms

Bootstrap (sampling with replacement) and subsampling (without replacement) can discern various statistics from replicate subsets of data while the samples in both cases are independent of each other. Our goal is to obtain a reliable clustering with measurable uncertainty from a set of different *k*-means partitions. The key idea of the approach is to integrate multiple partitions produced by clustering of pseudo-samples of a data set.

Clustering combinations can be formalized as follows. Let D be a data set of N data points in d-dimensional space. The input data can be represented as an $N \times d$ pattern matrix or $N \times N$ dissimilarity matrix, potentially in a non-metric space. Suppose that $X = \{X_1,\ldots,X_B\}$ is a set of B bootstrap samples of size N or subsamples of size $S < N$. A chosen clustering algorithm is run on each of the samples in X, which results in B partitions $\Pi=\{\pi_1, \pi_2,\ldots, \pi_B\}$. Each component partition in Π is a set of non-overlapping and exhaustive clusters with $\pi_i=\{C_1^i, C_2^i,\ldots,C_{k(i)}^i\}$, $X_i = C_1^i \cup \ldots \cup C_{k(i)}^i$, $\forall\pi_i$, where $k(i)$ is the number of clusters in the *i*-th partition.

The problem of combining partitions is to find a new partition $\sigma =\{C_1,\ldots,C_M\}$ of the entire data set D given the partitions in Π, such that the data points in any cluster of σ are more similar to each other than to points in different clusters within σ. We assume that the number of clusters, M, in the consensus clustering is predefined and can be different from the number of clusters, k, in the ensemble partitions. In order to find the target partition σ, one needs a consensus function utilizing information from the partitions $\{\pi_i\}$.

Several known consensus functions (Fred & Jain, 2002; Strehl & Ghosh, 2002; Topchy et al., 2003a) can be employed to map a given set of partitions Π={π_1, π_2,…, π_B} to the target partition, σ, in our study.

The similarity between two objects, x and y, is defined as follows:

$$sim(x,y)=\frac{1}{B}\sum_{i=1}^{B}\delta(\pi_i(x),\pi_i(y)),\quad \delta(a,b)=\begin{cases}1, & \text{if } a=b\\ 0, & \text{if } a\neq b\end{cases} \qquad (5.3)$$

Similarity between a pair of objects simply counts the number of clusters shared by the objects in the partitions {π_1,…, π_B}. Under the assumption that diversity comes from independent resampling, two families of algorithms can be proposed for integrating clustering components (Minaei-Bidgoli et al., 2004a,b).

5.3.1 Similarity-based algorithm

The first algorithm family is based on the co-association matrix, and employs a group of hierarchical clustering algorithms to find the final target partition. In this type, similarity-based clustering algorithms are used as the consensus function, Γ. Hierarchical clustering consensus functions with single-, complete-, and average-linkage criteria were used to obtain a target consensus clustering, σ. The pseudocode of these algorithms is shown in Figure 5.3.

```
Input: $D$ – the input data set $N$ points
$B$ – number of partitions to be combined
$M$ – number of clusters in the final partition, $\sigma$
$k$ – number of clusters in the components of the combination
$\Gamma$ – a similarity-based clustering algorithm
for $j$=1 to $B$
    Draw a random pseudosample $X_{\mathrm{j}}$
    Cluster the sample $X_{\mathrm{j}}$: $\pi(i) \leftarrow k$-means($\{X_{\mathrm{j}}\}$)
    Update similarity values (co-association matrix) for all patterns in $X_{\mathrm{j}}$
end
Combine partitions via chosen $\Gamma$: $\sigma \leftarrow \Gamma(P)$
Validate final partition, $\sigma$ (optional)
return $\sigma$ // consensus partition
```

Figure 5.3 First algorithms for clustering ensemble, based on co-association matrix and using different similarity-based consensus functions

5.3.2 Algorithms based on categorical clustering

The second family of algorithms for achieving clustering combination is based on new features extracted through the partitioning process. In this approach, one can view consensus clustering as clustering in a space of new features induced by the set, Π. Each partition, π_i, represents a feature vector with categorical values. The cluster labels of each object in different partitions are treated as a new feature vector, a B-tuple, given B different partitions in Π.

Table 5.1 (a) Data points and feature values, *N* rows and *d* columns. Every row of this table shows a feature vector corresponding to *N* points. (b) Partition labels for resampled data, *n* rows and *B* columns.

(a)

Data	**Features**					
$\mathbf{X_1}$	x_{11}	x_{12}	…	x_{1j}	…	x_{1d}
$\mathbf{X_2}$	x_{21}	x_{22}	…	x_{2j}	…	x_{2d}
…	…	…	…	…	…	…
$\mathbf{X_i}$	x_{i1}	x_{i2}	…	x_{ij}	…	x_{id}
…	…	…	…	…	…	…
$\mathbf{X_N}$	x_{N1}	x_{N2}	…	x_{Nj}	…	x_{Nd}

(b)

Data	**Partitions Labels**					
$\mathbf{X_1}$	$\pi_1(x_1)$	$\pi_2(x_1)$	…	$\pi_j(x_1)$	…	$\pi_B(x_1)$
$\mathbf{X_2}$	$\pi_1(x_2)$	$\pi_2(x_2)$	…	$\pi_j(x_2)$	…	$\pi_B(x_2)$
…	…	…	…	…	…	…
$\mathbf{X_i}$	$\pi_1(x_i)$	$\pi_2(x_i)$	…	$\pi_j(x_i)$	…	$\pi_B(x_i)$
…	…	…	…	…	…	…
$\mathbf{X_N}$	$\pi_1(x_N)$	$\pi_2(x_N)$	…	$\pi_j(x_N)$	…	$\pi_B(x_N)$

Therefore, instead of the original *d* attributes, which are shown in Table 5.1(a), the new feature vectors from a table with *N* rows and *B* columns (Table 5.1(b)) are utilized, where each column corresponds to the results of mapping a clustering algorithm (*k*-means) onto the resampled data and every row is a new feature extracted vector, with categorical (nominal) values. Here, $\pi_j(x_i)$ denotes the label of object x_i in the *j*-th partition of Π. Hence the problem of combining partitions becomes a categorical clustering problem.

```
Input: $D$ – the input data set $N$ points
$B$ - number of partitions to be combined
$M$ – number of clusters in the final partition $\sigma$
$k$ – number of clusters in the components of the combination
$\Gamma$ - consensus function operating with categorical features
Reference Partition $\leftarrow$ $k$-means($D$)
for $i$=1 to $B$
    Draw a random pseudo-sample $X_j$
    Cluster the sample $X_j$: $\pi(i) \leftarrow k$-means($\{X_j\}$)
    Store partition $\pi_i$
end
Re-label (if necessary)
Apply consensus function $\Gamma$ on the set of partition labels, $\Pi$, to find final partition $\sigma$
Validate final partition $\sigma$ (optional)
return $\sigma$ // consensus partition
```

Figure 5.4 Algorithms for clustering ensemble based on categorical clustering

The parameter *k* in both algorithms is the number of clusters in every component partition. If the value of *k* is too large then the partitions will overfit the data set, and if *k* is too small then the number of clusters may not be large enough to capture the true structure of data set. In addition, if the total number of clusterings, *B,* in the combination is too small then the effective sample size for the estimates of distances between co-association values is also insufficient, resulting in a larger variance. In the case of the subsampling algorithm (without replacement), the right choice of sample size *S* is closely related to the value of *k* and the value of *B* and proper setting of *S* is required to reach convergence to the true structure of the data set. The algorithmic parameters will be discussed in section 6. In the rest of this chapter "*k"* stands for number of clusters in every partition, "*B*" for number of partitions/pseudosamples (in both the bootstrap and the subsampling algorithms), and *"S"* for the sample size.

5.4 Consensus functions

A consensus function maps a given set of partitions $\Pi = \{\pi_1,\ldots, \pi_B\}$ to a target partition σ. In this experiment we have employed four types of consensus functions:

5.4.1 Co-association based functions

This consensus function operates on the co-association matrix. Similarity between points (co-association values) can be estimated by the number of clusters shared by two points in all the partitions of an ensemble. Then, numerous hierarchical agglomerative algorithms (criteria) can be applied to the co-association matrix to obtain the final partition, including Single Link (SL), Average Link (AL) and Complete Link (CL) (Jain & Dubes, 1988). There are three main drawbacks to this approach.

- First, it has a quadratic computational complexity in the number of patterns and features $O(kN^2d^2)$ (Duda et al., 2001), where k is the number of clusters, N is the number of data points, and d is the number of features.
- Second, there are no established guidelines for which clustering algorithm should be applied, e.g. single linkage or complete linkage.
- Third, an ensemble with a small number of partitions may not provide a reliable estimate of the co-association values (Topchy et al. 2003b).

5.4.2 Quadratic Mutual Information Algorithm (QMI)

Assuming that the partitions are independent, a consensus function based on k-means clustering in the space of standardized features can effectively maximize a generalized definition of mutual information (Topchy et al., 2003a). The complexity of this consensus function is $O(kNB)$, where k is the number of clusters, N is the number of items, and *B* is the number of partitions. Though the QMI algorithm can be potentially trapped in a local optimum, its relatively low computational complexity allows the use of multiple restarts in order to choose a quality consensus solution with minimum intra-cluster variance.

5.4.3 Hypergraph partitioning

The clusters could be represented as hyperedges on a graph whose vertices correspond to the data points to be clustered. The problem of consensus clustering is then reduced to finding the minimum-cut of a resulting hypergraph. The minimum *k*-cut of this hypergraph into *k* components gives the required consensus partition (Strehl & Ghosh, 2002). Hypergraph algorithms seem to work effectively for approximately balanced clusters. Though the hypergraph partitioning problem is NP-hard, efficient heuristics to solve the *k*-way min-cut partitioning problem are known, i.e. the complexity of CSPA, HGPA and MCLA is estimated in Strehl & Ghosh (2002) as $O(kN^2B)$, $O(kNB)$, and $O(k^2NB^2)$, respectively. These hypergraph algorithms are described in Strehl & Ghosh (2002) and their corresponding source codes are available at http://www.strehl.com. A drawback of hypergraph algorithms is that they seem to work the best for nearly balanced clusters (Topchy et al., 2003b).

5.4.4 Voting approach

In the previous algorithms there is no need to explicitly solve the correspondence problem between the labels of known and derived clusters. The voting approach attempts to solve the correspondence problem and then uses a majority vote to determine the final consensus partition (Dudoit & Fridlyand, 2003). The main idea is to permute the cluster labels such that the best agreement between the labels of two partitions is obtained. All the partitions from the ensemble must be re-labeled according to a fixed reference partition. The complexity of this process is $O(k!)$, which can be reduced to $O(k^3)$ if the Hungarian method is employed for the equivalent minimal weight bipartite matching problem.

All the partitions in the ensemble can be re-labeled according to their best agreement with some chosen reference partition. A meaningful voting procedure assumes that the number of clusters in every given partition is the same as in the target partition. This requires that the number of clusters in the target consensus partition is known (Topchy et al. 2003b).

The performance of all these consensus methods is empirically analyzed as a function of two important parameters: the type of sampling process (sample redundancy) and the granularity of each partition (number of clusters).

5.5 Critical issues in resampling

Let us emphasize the challenging points of using resampling techniques for maintaining diversity of partitions and estimation of co-association values.

5.5.1 Variable number of samples

There is no essential difference between bootstrap and subsampling algorithms with regard to the diversity of the data samples. In both cases the pseudosample can be incomplete (missing some objects). In bootstrap one has no control over the number of distinct objects in a sample, while in subsampling the size of a pseudosample can be much smaller than the original sample size. On the average, 37% of objects are not included into a bootstrap sample (Hastie et al., 2001).

5.5.2 Repetitive data points (objects)

In resampling with replacement (bootstrap) some of the data points can be drawn multiple times. However, in computing the co-association matrix, only one copy of a data point should contribute to the co-association similarity. Hence, for repeated objects, we count each pair of objects only once. This problem does not appear in the case of subsampling

5.5.3 Similarity estimation

Co-association values require adjustment when bootstrap partitions are used. Typically the similarity value between the objects *x* and *y, sim*(*x,y*), is calculated by counting the number of shared clusters in all the given partitions (Eq. 5.4). This is justified when each possible pair of objects appears the same number of times in these partitions. However, in bootstrap, different objects can appear in a different number of samples. Therefore, the effective sample size for a pair of objects may no longer be equal to *B*. Hence, co-association values should be computed as following:

$$sim\ (x, y) = \frac{1}{R}\sum_{i=1}^{R} \delta\,(P_i(x),\ P_i(y))\ , \qquad (5.4)$$

where *R* is the number of bootstrap samples containing both *x* and *y*, and the sum is taken over such samples.

5.5.4 Missing labels

In both bootstrap and subsampling, some of the objects are missed in drawn samples. When one uses co-association based methods, this poses no difficulty because the co-association values are only updated for existing objects. However, missing labels can cause a number of problems for other consensus functions. For example, when an object is missing in a bootstrap sample, there will be no label assigned to it after running the clustering algorithm. Thus, special consideration of the missing labels is necessary during the process of re-labeling, before running a consensus function.

5.5.5 Re-labeling

We must consider how to re-label two bootstrap samples with missing values. When the number of objects in the drawn samples is too small, this problem becomes harder. For example, consider four partitions, $P_{1, \ldots,}P_4$ for five data points $x_{1,}$ …,x_5 as shown in Table 5.2.

Table 5.2 An illustrative example of re-labeling difficulty involving five data points and four different clusterings of four bootstrap samples. The numbers represent the labels assigned to the objects and the "?" shows the missing labels of data points in the bootstrapped samples.

	P_1	P_2	P_3	P_4
x_1	1	?	2	3
x_2	1	2	3	1
x_3	?	2	?	2
x_4	?	?	1	?
x_5	2	1	3	1

One can re-label the above partitions in relation to some reference partition. However, the missing labels should not be considered in the re-labeling process. Therefore, if the reference partition is P_1, and we want to re-label P_2, then only the data points x_2 and x_5 participate in the re-labeling process. Similarly, if P_3 is re-labeled based on the reference P_1, then x_1, x_2 and x_3 are used in the Hungarian algorithm to find the best match. Once the best agreement among the given labels is found then all the objects in the partition, except those with missing labels, are re-labeled.

5.5.6 Adaptation of the k-means algorithm

The core of the QMI consensus function is the *k*-means algorithm, which utilizes a special transformation of the space of $N \times B$ extracted labels. Since no labels are assigned to the missing objects using the clustering algorithm, the missing objects should not be considered in the *k*-means algorithm that calculates the mutual information among the obtained labels and the target partition. A revised and extended version of the *k*-means algorithm was developed such that it can handle the missing coordinates, as described in Jain & Dubes (1988) and Dixon (1979). The *k*-means algorithm calculates the Euclidean distances between every object and the cluster centers. If some coordinates are missing then those coordinates are ignored in the calculation. The experiments in this study used this modification of the *k*-means for the QMI algorithm.

5.6 Adaptive sampling scheme

While there are many ways to construct diverse data partitions for an ensemble, not all of them easily generalize to adaptive clustering. The adaptive approach (Topchy et al., 2004) extends the studies of ensembles whose partitions are generated via data resampling (Dudoit & Fridlyand 2003; Fisher & Buhmann, 2003; Minaei et al, 2004). Though, intuitively, clustering ensembles generated by other methods also can be boosted. The adaptive partition generation mechanism (Brieman, 1998) is aimed at reducing the variance of inter-class decision boundaries. Unlike the regular bootstrap method that draws subsamples uniformly from a given data set, adaptive sampling favors points from regions close to the decision boundaries. At the same time, the points located far from the boundary regions are sampled less frequently. It is instructive to consider a simple example that shows the difference between ensembles of bootstrap partitions with and without the weighted sampling. Figure 5.5 shows how different decision boundaries can separate two natural classes depending on the sampling probabilities. Here we assume that the *k*-means clustering algorithm is applied to the subsamples.

Initially, all the data points have the same weight, namely, the sampling probability $p_i = \frac{1}{N}$, $i \in [1,\ldots,N]$. Clearly, the main contribution to the clustering error is due to the sampling variation that causes inaccurate inter-cluster boundaries. Solution variance can be significantly reduced if sampling is increasingly concentrated only on the subset of objects at iterations $t_2 > t_1 > t_0$, as demonstrated in Figure 5.5.

The key issue in the design of the adaptation mechanism is the updating of probabilities. We have to decide how and which data points should be sampled as we collect more and more clusterings in the ensemble. A consensus function based on the co-association values (Jain & Fred, 2002) provides the necessary guidelines for adjustments of sampling probabilities. Remember that the co-association similarity between two data points, *x* and *y*, is defined as the number of clusters shared by these points in the partitions of an ensemble, Π:

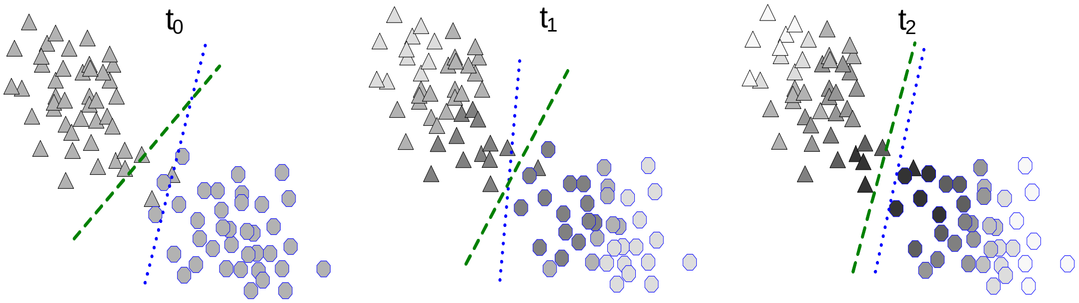
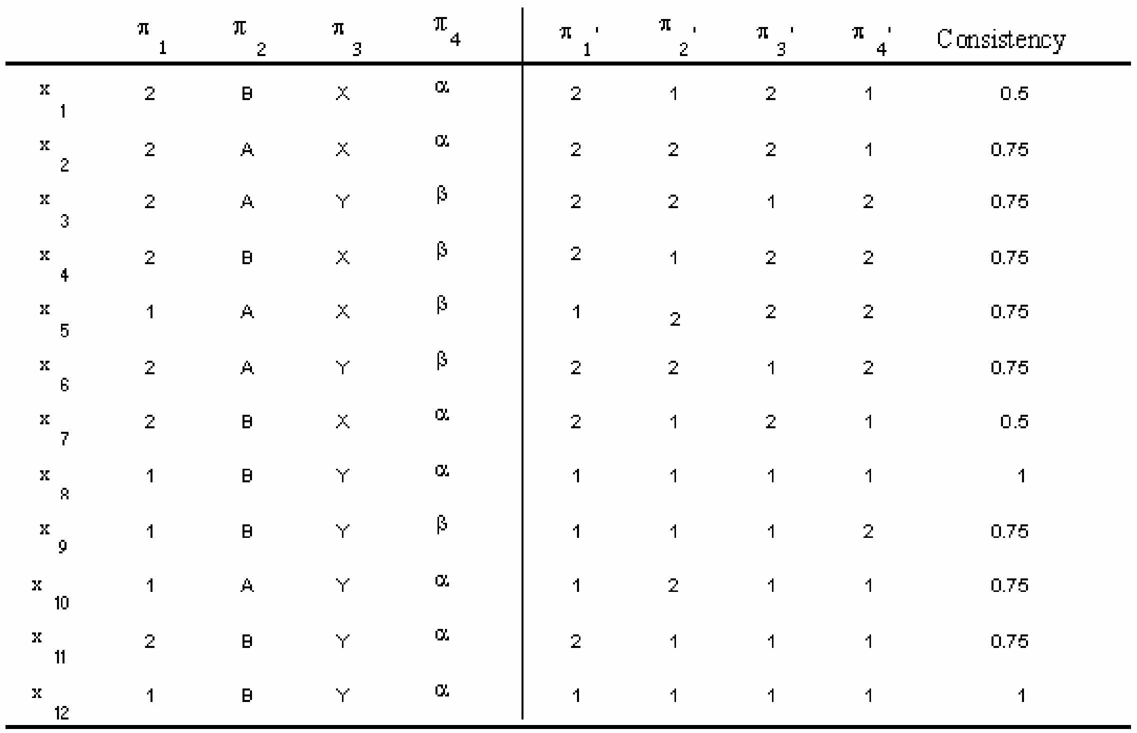


Figure 5.5 Two possible decision boundaries for a 2-cluster data set. Sampling probabilities of data points are indicated by gray level intensity at different iterations ($t_0 < t_1 < t_2$) of the adaptive sampling. True components in the 2-class mixture are shown as circles and triangles.

Table 5.3. Consistent re-labeling of 4 partitions of 12 objects.



	π_1	π_2	π_3	π_4	π_1'	π_2'	π_3'	π_4'	Consistency
x_1	2	B	X	α	2	1	2	1	0.5
x_2	2	A	X	α	2	2	2	1	0.75
x_3	2	A	Y	β	2	2	1	2	0.75
x_4	2	B	X	β	2	1	2	2	0.75
x_5	1	A	X	β	1	2	2	2	0.75
x_6	2	A	Y	β	2	2	1	2	0.75
x_7	2	B	X	α	2	1	2	1	0.5
x_8	1	B	Y	α	1	1	1	1	1
x_9	1	B	Y	β	1	1	1	2	0.75
x_{10}	1	A	Y	α	1	2	1	1	0.75
x_{11}	2	B	Y	α	2	1	1	1	0.75
x_{12}	1	B	Y	α	1	1	1	1	1

A consensus clustering can be found by using an agglomerative clustering algorithm (e.g., single linkage) applied to such a co-association matrix constructed from all the points. The quality of the consensus solution depends on the accuracy of similarity values as estimated by the co-association values. The least reliable co-association values come from the points located in the problematic areas of the feature space. Therefore, our adaptive strategy is to increase the sampling probability for such points as we proceed with the generation of different partitions in the ensemble.

The sampling probability can be adjusted not only by analyzing the co-association matrix, which is of quadratic complexity $O(N^2)$, but also by applying the less expensive $O(N + K^3)$ estimation of clustering consistency for the data points. Again, the motivation is that the points with the least stable cluster assignments, namely those that frequently change the cluster they are assigned to, require an increased presence in the data subsamples. In this case, a label correspondence problem must be approximately solved to obtain the same labeling of clusters throughout the ensemble's partitions. By default, the cluster labels in different partitions are arbitrary. To make the correspondence problem more tractable, one needs to re-label each partition in the ensemble using some fixed reference partition. Table 5.3 illustrates how four different partitions of twelve points can be re-labeled using the first partition as a reference.

At the (t+1)-th iteration, when some t different clusterings are already included in the ensemble, we use the Hungarian algorithm for minimal weight bipartite matching problem in order to re-label the (t+1)-th partition. As an outcome of the re-labeling procedure, we can compute the consistency index of clustering for each data point. Clustering consistency index CI at iteration t for a point x is defined as the ratio of the maximal number of times the object is assigned in a certain cluster to the total number of partitions:

$$CI(x) = \frac{1}{B} \max \left\{ \sum_{i=1}^{B} \delta(\pi_i(x), L \right\}_{[L \in cluster\_labels]} \qquad (5.5)$$

The values of consistency indices are shown in Table 5.3 after four partitions were generated and re-labeled. We should note that clustering of subsamples of the data set, D, does not provide the labels for the objects missing (not drawn) in some subsamples. In this situation, the summation in Eq. (5.5) skips the terms containing the missing labels.

The clustering consistency index of a point can be directly used to compute its sampling probability. In particular, the probability value is adjusted at each iteration as follows:

$$p_{t+1}(x) = Z(\alpha\, p_t(x) + 1 - CI(x)), \qquad (5.6)$$

where α is a discount constant for the current sampling probability and *Z* is a normalization factor. The discount constant was set to α=0.3 in our experiments. The proposed clustering ensemble algorithm is summarized in pseudocode in Figure 5.6:

```
Input: $D$ – data set of $N$ points
$B$ – number of partitions to be combined
$M$ – number of clusters in the consensus partition $\sigma$
$K$ – number of clusters in the partitions of the ensemble
$\Gamma$ – chosen consensus function operating on cluster labels
$\mathbf{p}$ – sampling probabilities (initialized to $1/N$ for all the points)
Reference Partition ← $k$-means($D$)
for $i$=1 to $B$
    Draw a subsample $X_i$ from $D$ using sampling probabilities $\mathbf{p}$
    Cluster the sample $X_i$: $\pi(i) \leftarrow k$-means($X_i$)
    Re-label partition $\pi(i)$ using the reference partition
    Compute the consistency indices for the data points in $D$
    Adjust the sampling probabilities $\mathbf{p}$
end
Apply consensus function $\Gamma$ to ensemble $\Pi$ to find the partition $\sigma$
Validate the target partition $\sigma$ (optional)
return $\sigma$  // consensus partition
```

Figure 5.6 Algorithms for adaptive clustering ensembles

5.7 Experimental study on non-adaptive approaches

The experiments were performed on several data sets, including two challenging artificial problem, the “Halfrings” data set, and the “2-Spiral” data set, two data sets from UCI repository, the “Iris” and “Wine” and two other real world data set, the “LON” and “Star/Galaxy” data sets. A summary of data set characteristics is shown in Table 5.4.

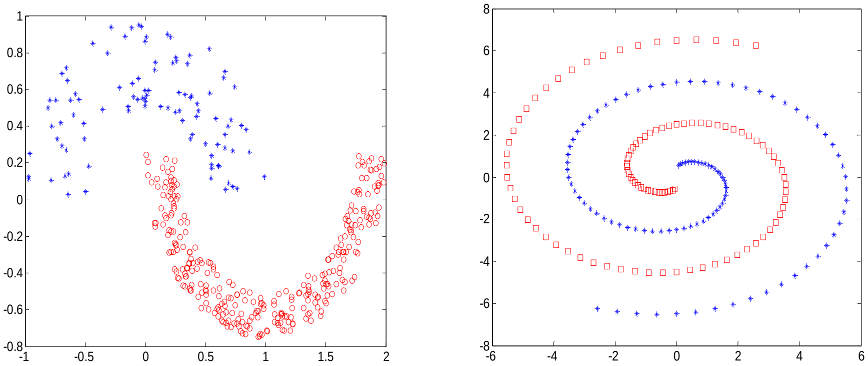


Figure 5.7 “Halfrings” data set with 400 patterns (100-300 per class) , “2-Spirals” dataset with 200 patterns (100-100 per class)

Table 5.4. A summary of data sets characteristics

	No. of Classes	***No. of Features***	***No. of Patterns***	***Patterns per class***
Star/Galaxy	2	14	4192	2082-2110
Wine	3	13	178	59-71-48
LON	2	6	227	64-163
Iris	3	4	150	50-50-50
3-Gaussian	3	2	300	50-100-150
Halfrings	2	2	400	100-300
2-Spirals	2	2	200	100-100

5.7.1 Data sets

The Halfrings and 2-Spiral data set, as shown in Figure 5.7, consist of two clusters, though the clusters are unbalanced with 100- and 300-point patterns in the Halfrings data set and balanced in the 2-Spiral. The *k*-means algorithm by itself is not able to detect the two natural clusters since it implicitly assumes hyperspherical clusters. 3-Gaussian is a simulated data set that includes three unbalanced classes with 50, 100, and 150 data points. The Wine data set described in Aeberhard et al. (1992) contains the value of the chemical composition of wines grown in the same region but derived from three different cultivars. The patterns are described by the quantities of thirteen constituents (features) found in each of the three types of wines. There are 178 samples in total.

The LON data set (Minaei & Punch, 2003) is extracted from the activity log in a web-based course using an online educational system developed at Michigan State University (MSU): the Learning Online Network with Computer-Assisted Personalized Approach (LON-CAPA). The data set includes the student and course information on an introductory physics course (PHY183), collected during the spring semester 2002. This course included 12 homework sets with a total of 184 problems, all of which were completed online using LON-CAPA. The data set consists of 227 student records from one of the two groups: "Passed" for the grades above 2.0, and "Failed" otherwise. Each sample contains 6 features.

The Iris data set contains 150 samples in 3 classes of 50 samples each, where each class refers to a type of iris plant. One class is linearly separable from the other two, and each sample has four continuous-valued features. The Star/Galaxy data set described in Odewahn (1992) has a significantly larger number of samples (N=4192) and features (d=14). The task is to separate observed objects into stars or galaxies. Domain experts manually provided true labels for these objects.

For all these data sets the number of clusters, and their assignments, are known. Therefore, one can use the misassignment (error) rate of the final combined partition as a measure of performance of clustering combination quality. One can determine the error rate after solving the correspondence problem between the labels of derived and known clusters. The Hungarian method for solving the minimal weight bipartite matching problem can efficiently solve this label correspondence problem.

5.7.2 The role of algorithm's parameters

The bootstrap experiments probe the accuracy of partition combination as a function of the resolution of partitions (value of *k*) and the number of partitions, *B* (number of partitions to be merged).

One of our goals was to determine the minimum number of bootstrap samples, *B*, necessary to form high-quality combined cluster solutions. In addition, different values of *k* in the *k*-means algorithm provide different levels of resolution for the partitions in the combinations. We studied the dependence of the overall performance on the number of clusters, *k*. In particular, clustering on the bootstrapped samples was performed for the values of *B* in the range [5, 1000] and the values of *k* in the interval [2, 20].

Analogously, the size of the pseudosample, *S*, in subsampling experiments is another important parameter. Our experiments were performed with different subsample sizes in the interval [*N*/20, 3*N*/4], where *N* is the size of the original data sample. Thus, in the case of the Halfrings, *S* was taken in the range [20, 300] where the original sample size is *N*=400, while in the case of the Galaxy data set, parameter *S* was varied in the range [200, 3000] where *N*=4192. Therefore, in resampling without replacement, we analyzed how the clustering accuracy was influenced by three parameters: number of clusters, *k*, in every clustering, number of drawn samples, *B*, and the sample size, *S.* Note that all the experiments were repeated 20 times and the average error rate for 20 independent runs is reported, except for the Star/Galaxy data where 10 runs were performed.

The experiments employed eight different consensus functions: co-association based functions (single link, average link, and complete link), hypergraph algorithms (HGPA, CSPA, MCLA), the QMI algorithm, as well as a Voting-based function.

5.7.3 The Role of Consensus Functions (Bootstrap algorithm)

Perhaps the single most important design element of the combination algorithm is the choice of a consensus function. In the Halfrings data set the true structure of the data set (100% accuracy) was obtained using co-association based consensus functions (both single and average link) in the case of *k*=15 and number of partitions taking part in the combination where *B*≥100. None of the other six consensus methods converged to an acceptable error rate for this data set.

For the Wine data set an optimal accuracy of 73% was obtained with both the hypergraph-CSPA algorithm and co-association based method using average link (AL) with different parameters as shown in Table 5.6. For the LON data set the optimal accuracy of 79% was achieved only by co-association-based (using the AL algorithm) consensus function. This accuracy is comparable to the results of the *k*-NN classifier, multilayer perceptron, naïve Bayes classifier, and some other algorithms when the "*LON*" data set is classified in a supervised framework based on labeled patterns (Minaei & Punch, 2003).

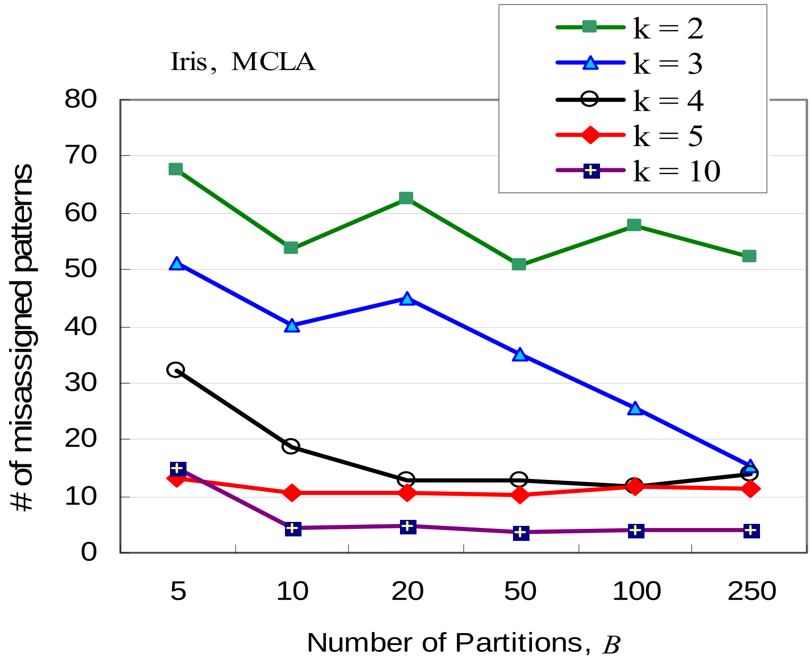


Figure 5.8 “Iris” data set. Bootstrapping for fixed consensus function MCLA, different *B*, and different values of *k*.

For the “*Iris*” data set, the hypergraph consensus function, HPGA algorithm led to the best results when k ≥ 10. The AL and the QMI algorithms also gave acceptable results, while the single link and average link did not demonstrate a reasonable convergence. Figure 5.8 shows that the optimal solution could not be found for the Iris data set with *k* in the range [2, 5], while the optimum was reached for *k* ≥ 10 with only *B*≥10 partitions.

For the Star/Galaxy data set the CSPA function (similarity based hypergraph algorithm) could not be used due to its computational complexity because it has a quadratic complexity in the number of patterns $O(kN^2B)$.

The HGPA function and SL did not converge at all, as shown in Table 5.5. Voting and complete link also did not yield optimal solutions. However, the MCLA, the QMI and the AL functions led to an error rate of approximately 10%, which is better than the performance of an individual *k*-means result (21%).

The major problem in co-association based functions is that they are computationally expensive. The complexity of these functions is very high ($O(kN^2d^2)$) and therefore, it is not effective to use the co-association based functions as a consensus function for the large data sets.

Table 5.5 "Star/Galaxy" data experiments. Average error rate (% over 10 runs) of clustering combination using resampling algorithms with different number of components in combination B, resolutions of components, k, and types of consensus functions.

K	B	QMI	MCLA	SL	AL	CL	Voting
2	5	18.5	19.4	49.7	49.7	49.7	20.4
2	10	18.7	18.8	49.6	49.6	49.6	19.5
2	20	18.5	18.9	49.6	24.4	49.7	19
2	50	18.7	18.8	49.6	18.8	49.7	18.9
2	100	18.8	18.8	49.7	18.8	18.8	18.9
3	5	13.4	15.5	49.7	49.7	49.7	-
3	10	17.8	15.6	49.6	49.6	49.6	-
3	20	11.5	15.3	49.7	18.8	42.9	-
3	50	13.3	15.4	49.7	11	35.9	-
3	100	11	15.4	49.7	11	48.2	-
4	5	15.2	13.1	49.7	49.7	49.7	-
4	10	11.4	14.5	49.6	49.7	49.7	-
4	20	14	13.7	49.6	24.3	48.7	-
4	50	22.2	11.9	49.7	10.7	48	-
4	100	11	11.9	49.7	10.7	47.9	-
5	5	14.9	13.8	49.7	49.7	49.7	-
5	10	14.9	13.1	49.7	47.9	49.6	-
5	20	10.7	13.4	49.6	11	49.7	-
5	50	11.4	13.4	49.7	10.8	48.7	-
5	100	11	12.5	49.7	10.9	48	-

Note that the QMI algorithm did not work well when the number of partitions exceeded 200, especially when the value of *k* was large. This might be due to the fact that the core of the QMI algorithm operates in $k \times B$–dimensional space. The performance of the *k*-means algorithm degrades considerably when *B* is large (>100) and, therefore, the QMI algorithm should be used with smaller values of *B*.

5.7.4 Effect of the Resampling method (Bootstrap vs. Subsampling)

In subsampling the smaller the *S* the lower the complexity of the *k*-means clustering, which therefore results in much smaller complexity in the co-association based consensus functions, which is super-linear *N*. Comparing the results of the bootstrap and the subsampling methods shows that when the bootstrap technique converges to an optimal solution, that optimal result could be obtained by the subsampling as well, but with a critical size of the data points. For example, in the Halfrings data set the perfect clustering can be obtained using a single-link consensus function with *k*=10, *B*=100 and *S*=200 (1/5 data size) as shown in Figure 5.9 (compare to the bootstrap results in table 5.6) while this perfect results can be achieved with *k*=15, *B* = 50, and *S* = 80 (1/5 data size). Thus, there is a trade off between the number of partitions *B* and the sample size *S*. This comparison shows that the subsampling method can be much faster than the bootstrap (*N*=400) in relation to the computational complexity.

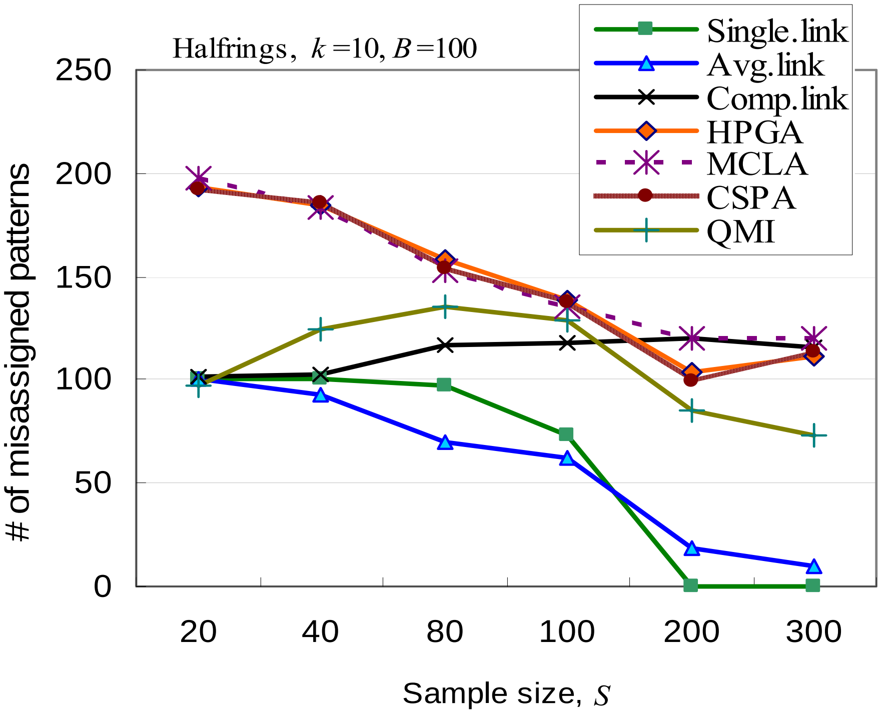


Figure 5.9 “Halfrings” data set. Experiments using subsampling with *k*=10 and *B*=100, different consensus function, and sample sizes *S*.

The results of subsampling for "Star/Galaxy" data set as shown in Figure 5.10, shows that in resolution *k*=3 and number of partitions *B*=100, with only sample size *S* = 500 (1/8 of the entire data size) one can reach 89% accuracy, the same results with entire data set in the bootstrap method. It shows that for this large data set, a small fraction of data can be representative of the entire data set, and this would be computationally very interesting in distributed data mining.

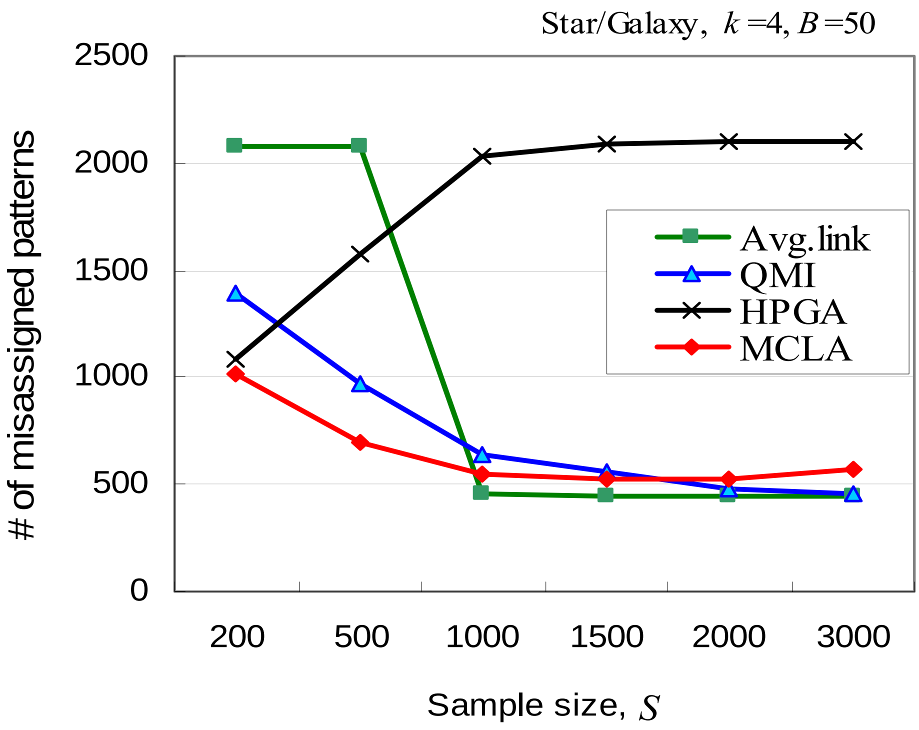


Figure 5.10 "Star/Galaxy" data set. Experiments using subsampling, with *k* = 4 and *B* = 50 and different consensus function and sample sizes *S*.

Note that in both the bootstrap and the subsampling algorithms all of the samples are drawn independently, and thus the resampling process could be performed in parallel. Therefore, using the *B* parallel processes, the computational process becomes *B* times faster.

Table 5.6 shows the error rate of classical clustering algorithms, which are used in this research. The error rates for the *k*-means algorithm were obtained as the average over 100 runs, with random initializations for the cluster centers, where value of *k* was fixed to the true number of clusters. One can compare it to the error rate of ensemble algorithms in Table 5.7.

Table 5.6 The average error rate (%) of classical clustering algorithms. An average over 100 independent runs is reported for the k-means algorithms

Data set	k-means	Single Link	Complete Link	Average Link
Halfrings	25%	24.3%	14%	5.3%
Iris	15.1%	32%	16%	9.3%
Wine	30.2%	56.7%	32.6%	42%
LON	27%	27.3%	25.6%	27.3%
Star/Galaxy	21%	49.7%	44.1%	49.7%

Table 5.7 Summary of the best results of Bootstrap methods

Data set	**Best Consensus function(s)**	**Lowest Error rate obtained**	**Parameters**
Halfrings	Co-association, SL	0%	k ≥ 10, B. ≥ 100
	Co-association, AL	0%	k ≥ 15, B ≥ 100
Iris	Hypergraph-HGPA	2.7%	k ≥ 10, B ≥ 20
Wine	Hypergraph-CSPA	26.8%	k ≥ 10, B ≥ 20
	Co-association, AL	27.9%	k ≥ 4, B ≥ 100
LON	Co-association, CL	21.1%	k ≥ 4, B ≥ 100
Galaxy/ Star	Hypergraph-MCLA	9.5%	k ≥ 20, B ≥ 10
	Co-association, AL	10%	k ≥ 10, B ≥ 100
	Mutual Information	11%	k ≥ 3, B ≥ 20

Table 5.8 Subsampling methods: trade-off among the values of *k*, the number of partitions *B*, and the sample size, *S*. Last column denote the percentage of sample size regarding the entire data set. (Bold represents most optimal)

Data set	**Best Consensus function(s)**	**Lowest Error rate**	***k***	***B***	***S***	**% of entire data**
Halfrings	SL	0%	10	100	200	50%
	SL	0%	10	500	80	**20%**
	AL	0%	15	1000	80	20%
	AL	0%	20	500	100	25%
Iris	HGPA	2.3%	10	100	50	33%
	HGPA	2.1%	15	50	50	**33%**
Wine	AL	27.5%	4	50	100	56%
	HPGA	28%	4	50	20	**11%**
	CSPA	27.5%	10	20	50	28%
LON	CL	21.5%	4	500	100	44%
	CSPA	21.3%	4	100	100	**44%**
Galaxy/ Star	MCLA	10.5%	10	50	1500	36%
	MCLA	11.7%	10	100	200	**5%**
	AL	11%	10	100	500	12%

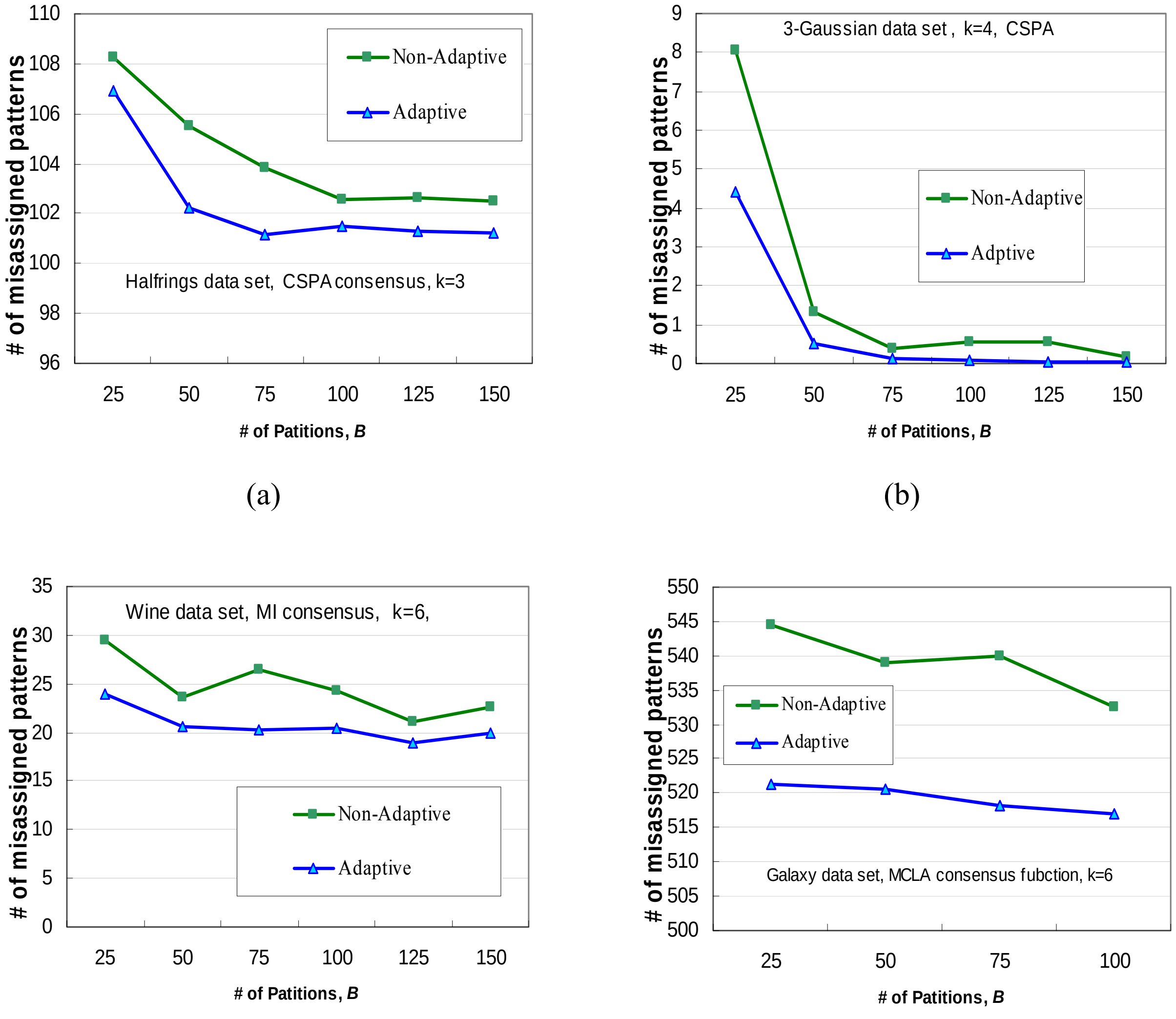
The optimal size *S* and granularity of the component partitions derived by subsampling are reported in Table 5.8. We see that the accuracy of the resampling method is very similar to that of the bootstrap algorithm, as reported in Table 5.6. This level of accuracy was reached with remarkably smaller sample sizes and much lower computational complexity! The trade-off between the accuracy of the overall clustering combination and computational effort for generating component partitions is shown in table 8, where we compare accuracy of consensus partitions. The most promising result is that only a small fraction of data (i.e., 12% or 5% for the “Star/Galaxy” data set) is required to obtain the optimal solution of clustering, both in terms of accuracy and computational time.

The question of the best consensus function remains open for further study. Each consensus function explores the structure of data set in different ways, thus its efficiency greatly depends on different types of existing structure in the data set. One can suggest having several consensus functions and then combining the consensus function results through maximizing mutual information (Strehl & Ghosh; 2002), but running different consensus functions on large data sets would be computationally expensive.

5.8 Empirical study and discussion of Adaptive approach

The experiments were conducted on artificial and real-world data sets ("Galaxy", "half-rings", "wine", "3-gaussian", "Iris", "LON"), with known cluster labels, to validate the accuracy of consensus partition. A comparison of the proposed adaptive and previous non-adaptive (Minaei et al. 2004) ensemble is the primary goal of the experiments. We evaluated the performance of the clustering ensemble algorithms by matching the detected and the known partitions of the datasets. The best possible matching of clusters provides a measure of performance expressed as the misassignment rate. To determine the clustering error, one needs to solve the correspondence problem between the labels of known and derived clusters. Again, the Hungarian algorithm was used for this purpose. The *k*-means algorithm was used to generate the partitions of samples of size *N* drawn with replacement, similar to bootstrap, albeit with dynamic sampling probability. Each experiment was repeated 20 times and average values of error (misassignment) rate are shown in Figure 5.11.

Consensus clustering was obtained by four different consensus functions: hypergraph-based MCLA and CSPA algorithms (Strehl & Ghosh; 2002), quadratic mutual information (Topchy et al., 2003a) and EM algorithm based on mixture model (Topchy et al., 2003b). Herein, we report only the key findings. The main observation is that adaptive ensembles slightly outperform the regular sampling schemes on most benchmarks. Typically, the clustering error decreased by 1-5%. Accuracy improvement depends on the number of clusters in the ensemble partitions (*k*). Generally, the adaptive ensembles were superior for values of *k* larger than the target number of clusters, *M*, by 1or 2. With either too small or too large a value of *k*, the performance of adaptive ensembles was less robust and occasionally worse than corresponding non-adaptive algorithms. A simple inspection of probability values always confirmed the expectation that points with large clustering, uncertainty are drawn more frequently.



(a)

(b)

(c)

(d)

Figure 5.11 Clustering accuracy for ensembles with adaptive and non-adaptive sampling mechanisms as a function of ensemble size for some data sets and selected consensus functions.

Most significant progress was detected when combination consisted of 25-75 partitions. Large numbers of partitions (*B*>75) almost never lead to further improvement in clustering accuracy. Moreover, for *B*>125 we often observed increased error rates (except for the hypergraph-based consensus functions), due to the increase in complexity of the consensus model and in the number of model parameters requiring estimation.

5.9 Concluding remarks

A new approach to combine partitions is proposed by resampling of original data. This study showed that meaningful consensus partitions for the entire data set of objects emerge from clusterings of bootstrap and subsamples of small size. Empirical studies were conducted on various simulated and real data sets for different consensus functions, number of partitions in the combination and number of clusters in each component, for both bootstrap (*with* replacement) and subsampling (*without* replacement). The results demonstrate that there is a trade-off between the number of clusters per component and the number of partitions, and the sample size of each partition needed in order to perform the combination process converges to an optimal error rate.

The bootstrap technique was recently applied in (Dudoit & Fridlyand, 2003; Fisher & Buhmann, 2003; Monti et al., 2003) to create a diversity in clusterings ensemble. However, our work extends the previous studies by using a more flexible subsampling algorithm for ensemble generation. We also provided a detailed comparative study of several consensus techniques. The challenging points of using resampling techniques for maintaining diversity of partitions were discussed in this chapter. We showed that there exists a critical fraction of data such that the structure of entire data set can be perfectly detected. Subsamples of small sizes can reduce costs and measurement complexity for many explorative data mining tasks with distributed sources of data.

We have extended clustering ensemble framework by adaptive data sampling mechanism for generation of partitions. We dynamically update sampling probability to focus on more uncertain and problematic points by on-the-fly computation of clustering consistency. Empirical results demonstrate improved clustering accuracy and faster convergence as a function of the number of partitions in the ensemble.

Further study of alternative resampling methods, such as the balanced (stratified) and recentered bootstrap methods are critical for more generalized and effective results. This work has bee published in (Minaei et al., 2004a; Minaei et al. 2004b, Topchy et al. 2004).

Chapter 6 Association Analysis in LON-CAPA

A key objective of data mining is to uncover the hidden relationships among the objects in a data set. Web-based educational technologies allow educators to study how students learn and which learning strategies are most effective. Since LON-CAPA collects vast amounts of student profile data, data mining and knowledge discovery techniques can be applied to find interesting relationships between attributes of students, assessments, and the solution strategies adopted by students. This chapter focuses on the discovery of interesting contrast rules, which are sets of conjunctive rules describing interesting characteristics of different segments of a population. In the context of web-based educational systems, contrast rules help to identify attributes characterizing patterns of performance disparity between various groups of students. We propose a general formulation of contrast rules as well as a framework for finding such patterns. We apply this technique to the LON-CAPA system.

6.1 Introduction

This chapter investigates methods for finding interesting rules based on the characteristics of groups of students or assignment problems. More specifically, our research is guided and inspired by the following questions: Can we identify the different groups of students enrolled in a particular course based on their demographic data? Which attribute(s) best explain the performance disparity among students over different sets of assignment problems? Are the same disparities observed when analyzing student performance in different sections or semesters of a course?

We address the above questions using a technique called contrast rules. Contrast rules are sets of conjunctive rules describing important characteristics of different segments of a population. Consider the following toy example of 200 students who enrolled in an online course. The course provides online reading materials that cover the concepts related to assignment problems. Students may take different approaches to solve the assignment problems. Among these students, 109 students read the materials before solving the problems while the remaining 91 students directly solve the problems without reviewing the materials. In addition, 136 students eventually passed the course while 64 students failed. This information summarized in a 2 × 2 contingency table as shown in Table 6.1.

Table 6.1 A contingency table of student success vs. study habits for an online course

	Passed	Failed	Total
Review materials	95	14	109
Do not review	41	50	91
Total	136	64	200

The table shows that there are interesting contrasts between students who review the course materials before solving the homework problems and students who do not review the materials. The following contrast rules can be induced from the contingency table:

Review materials $\Rightarrow$ Passed, $s = 47.5\%$, $c = 87.2\%$

Review materials $\Rightarrow$ Failed, $s = 7.0\%$, $c = 12.8\%$

Figure 6.1 A contrast rule extracted from Table 6.1

where s and c are the support and confidence of the rules (Agrawal et al., 1993). These rules suggest that students who review the materials are more likely to pass the course. Since there is a large difference between the support and confidence of both rules, the observed contrast is potentially interesting. Other examples of interesting contrast rules obtained from the same contingency table are shown in Figures 6.2 and 6.3.

Passed ⇒ Review materials,	$s = 47.5\%$,	$c = 69.9\%$
Failed ⇒ Review materials,	$s = 7.0\%$,	$c = 15.4\%$

Figure 6.2 A contrast rule extracted from Table 6.1

Passed ⇒ Review materials,	$s = 47.5\%$,	$c = 69.9\%$
Passed ⇒ Do not review,	$s = 20.5\%$,	$c = 30.1\%$

Figure 6.3 A contrast rule extracted from Table 6.1

Not all contrasting rule pairs extracted from Table 6.1 are interesting, as the example in Figure 6.4 shows.

Do not review ⇒ Passed,	$s = 20.5\%$,	$c = 45.1\%$
Do not review ⇒ Failed,	$s = 25.0\%$,	$c = 54.9\%$

Figure 6.4 A contrast rule extracted from Table 6.1

The above examples illustrate some of the challenging issues concerning the task of mining contrast rules:

1. There are many measures applicable to a contingency table. Which measure(s) yield the most significant/interesting contrast rules among different groups of attributes?

2. Many rules can be extracted from a contingency table. Which pair(s) of rules should be compared to define an interesting contrast?

This chapter presents a general formulation of contrast rules and proposes a new algorithm for mining interesting contrast rules. The rest of this chapter is organized as follows: Section 6.2 provides a brief review of related work. Section 6.3 offers a formal definition of contrast rules. Section 6.4 gives our approach and methodology to discover the contrast rules. Section 6.5 describes the LON-CAPA data model and an overview of our experimental results.

6.2 Background

In order to acquaint the reader with the use of data mining in online education, we present a brief introduction of association analysis and measures for evaluating association rules. Next, we explain the history of data mining in web-based educational systems. Finally, we discuss previous work related to contrast rules.

6.2.1 Association analysis

Let $I = \{i_1, i_2, \ldots, i_m\}$ be the set of all items and $T = \{t_1, t_2, \ldots, t_N\}$ the set of all transactions where m is the number of items and N is the number of transactions. Each transaction t_j is a set of items such that $t_j \subseteq I$. Each transaction has a unique identifier, which is referred to as TID. An *association rule* is an implication statement of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and X and Y are disjoint, that is, $X \cap Y = \varnothing$. X is called the antecedent while Y is called the consequence of the rule (Agrawal et al., 1993; Agrawal & Srikant, 1994).

Support and confidence are two metrics, which are often used to evaluate the quality and interestingness of a rule. The rule $X \Rightarrow Y$ has support, *s*, in the transaction set, *T*, if *s*% of transactions in *T* contains $X \cup Y$. The rule has *confidence*, *c*, if *c*% of transactions in *T* that contain *X* also contains *Y*. Formally, support is defined as shown in Eq. (6.1),

$$s(X \Rightarrow Y) = \frac{s(X \cup Y)}{N}, \tag{6.1}$$

where *N* is the total number of transactions, and confidence is defined in Eq. (6.2).

$$c(X \Rightarrow Y) = \frac{s(X \cup Y)}{s(X)}, \tag{6.2}$$

Another measure that could be used to evaluate the quality of an association rule is presented in Eq. (6.3).

$$RuleCoverage = \frac{s(X)}{N} \tag{6.3}$$

This measure represents the fraction of transactions that match the left hand side of a rule.

Techniques developed for mining association rules often generate a large number of rules, many of which may not be interesting to the user. There are many measures proposed to evaluate the interestingness of association rules (Freitas, 1999; Meo, 2003). Silberschatz and Tuzhilin (1995) suggest that interestingness measures can be categorized into two classes: objective and subjective measures.

An objective measure is a data-driven approach for evaluating interestingness of rules based on statistics derived from the observed data. In the literature different objective measures have been proposed (Tan et al., 2004). Examples of objective interestingness measure include support, confidence, correlation, odds ratio, and cosine.

Subjective measures evaluate rules based on the judgments of users who directly inspect the rules (Silberschatz & Tuzhilin, 1995). Different subjective measures have been addressed to discover the interestingness of a rule (Silberschatz & Tuzhilin, 1995). For example, a *rule template* (Fu & Han, 1995) is a subjective technique that separates only those rules that match a given template. Another example is *neighborhood*-based interestingness (Dong & Li, 1998), which defines a single rule's interestingness in terms of the supports and confidences of the group in which it is contained.

6.2.2 Data mining for online education systems

Recently, several researchers have worked on the application of data mining to examine or classify students' problem-solving approaches within web-based educational systems. For example, we previously developed tools for predicting the student performance with respect to average values of student attributes versus the overall problems of an online course (Minaei et al., 2003). Zaïane (2001) suggested the use of web mining techniques to build an agent that recommends on-line learning activities in a web-based course. Ma et al. (2000) focused on one specific task of using association rule mining to select weak students for remedial classes. This previous work focused on finding association rules with a specific rule consequent (i.e. a student is weak or strong). Herein, we propose a general formulation of contrast rules as well as a framework for finding such patterns.

6.2.3 Related work

An important goal in data mining is the discovery of major differences among segments of population. Bay and Pazzani (2001) introduced the notion of contrast sets as a conjunction of attributes and values that differ “meaningfully” in their distribution across groups. They used a chi-square test for testing the null hypothesis that contrast-set support is equal across all groups. They developed the STUCCO (Search and Testing for Understandable Consistent Contrast) algorithm to find significant contrast sets. Our work represents a general formulation for contrast rules using different interestingness measures. We show that alternative measures allow for different perspectives on the process of finding interesting rules.

Liu et al. (2001) have also used a chi-square test of independence as a principal measure for both generating the association rules and identifying non-actionable rules. Below, we briefly discuss the chi-square test of independence and one of its shortcomings.

Chi-square testing is used as a method for verifying the independence or correlation of attributes. The chi-square test compares observed frequencies with the corresponding expected frequencies. The greater the difference between observed and expected frequencies, the greater is the power of evidence in favor of dependence and relationship. Let *CT* be a contingency table with *K* rows and *L* columns. The chi-square test for independence is shown in Eq. (6.4) where $1 \leq i \leq K$, and $1 \leq j \leq L$, and degree of freedom is $(K-1)(L-1)$.

$$\chi^2 = \sum_i \sum_j \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \qquad (6.4)$$

However, a drawback of this test is that the χ^2 value is not invariant under the *row-column scaling* property (Tan et al., 2004). For example, consider the contingency table shown in Table 6.2(a). If χ^2 is higher than a specific threshold (e.g. 3.84 at the 95% significance level and degree of freedom 1), we reject the independence assumption. The chi-square value corresponding to Table 6.2(a) is equal to 1.82. Therefore, the null hypothesis is accepted. Nevertheless, if we multiply the values of that contingency table by 10, a new contingency table is obtained as shown in Table 6.2(b). The χ^2 value increases to 18.2 (>3.84). Thus, we reject the null hypothesis. We expect that the relationship between gender and success for both tables as being equal, even though the sample sizes are different. In general, this drawback shows that χ^2 is proportional to *N*.

Table 6.2 A contingency table proportional to table 6.1

(a)

	Passed	Failed	Total
Male	40	49	89
Female	60	51	111
Total	100	100	200

(b)

	Passed	Failed	Total
Male	400	490	890
Female	600	510	1110
Total	1000	1000	2000

6.3 Contrast Rules

In this section, we introduce the notion of contrast rules. Let *A* and *B* be two itemsets whose relationship can be summarized in a 2×2 contingency table as shown in Table 6.3.

Table 6.3 A contingency table for the binary case

	B	$\overline{B}$	Total
A	f_{11}	f_{12}	f_{1+}
$\overline{A}$	f_{21}	f_{22}	f_{2+}
Total	f_{+1}	f_{+2}	N

Let Ω be a set of all possible association rules that can be extracted from such a contingency table (Figure 6.5).

$$A \Rightarrow B\ ,\ A \Rightarrow \overline{B}\ ,\ \overline{A} \Rightarrow B\ ,\ \overline{A} \Rightarrow \overline{B}\ ,\ B \Rightarrow A\ ,\ \overline{B} \Rightarrow A\ ,\ B \Rightarrow \overline{A}\ ,\ \overline{B} \Rightarrow \overline{A}$$

Figure 6.5 Set of all possible association rules for Table 6.3.

We assume that *B* is a target variable and *A* is a conjunction of explanatory attributes. Let μ be a set of measures that can be applied to a rule or contingency table. Examples of such measures include support, confidence, chi-square, odds ratio, correlation, cosine, Jaccard, and interest (Tan et al., 2004). Below we provide a formal definition of “contrast rule.”

Definition (General Formulation of Contrast Rules):

A contrast rule, *cr,* is a 4-tuple <*br*, υ(*br*), *M*, Δ> where:

1. $br \subset \Omega$, is the base rule,
2. $\upsilon(br) \subset \Omega$ is a neighborhood to which the base rule *br* is compared,
3. M=<m_{base}, $m_{neighbor}$> is an ordered pair of measures where m_{base}, $m_{neighbor} \in \mu$, and m_{base} measures the rules in br and $m_{neighbor}$ measures the rules in υ(*br*),
4. $\Delta(m_{base}(br),\ m_{neighbor}(\upsilon(br)))$ is a comparison function between $m_{base}(r)$ and $m_{neighbor}(\upsilon(br))$.

A contrast rule, *cr,* is interesting if and only if $\Delta(m_{base}(br),\ m_{neighbor}(\upsilon(br))) \geq \sigma$, where σ is a user defined threshold, which implies that there is a large difference between *br* and its neighborhood with respect to *M*.

Figure 6.6 Formal definition of a contrast rule

As shown in Figure 6.6, the contrast rule definition is based on a paired set of rules, base rule *br* and its neighborhood υ(*br*). The base rule is a set of association rules with which a user is interested in finding contrasting association rules. Below are some examples that illustrate the definition.

Example 1*: cr$_1$ (Difference of confidence)*

The first type of contrast rules examines the difference between rules $A \Rightarrow B$ and $A \Rightarrow \overline{B}$. An example of this type of contrast was shown in Figure 6.1. Let confidence be the selected measure for both rules. Let absolute difference be the comparison function. We can summarize this type of contrast as follows:

- *br*: $\{A \Rightarrow B\}$
- υ(*r*): $\{A \Rightarrow \overline{B}\}$
- *M*: <confidence, confidence>
- Δ: absolute difference

The evaluation criterion for this example is shown in Eq. 6.5. This criterion can be used for ranking different pairs of contrast rules

$$\begin{aligned}\Delta &= |\, c(r) - c(\upsilon(r))\,| \\ &= |\, c(A \Rightarrow B) - c(A \Rightarrow \overline{B})\,| \\ &= \left|\frac{f_{11}}{f_{1+}} - \frac{f_{12}}{f_{1+}}\right| = \left|\frac{f_{11} - f_{12}}{f_{1+}}\right|, \end{aligned} \qquad (6.5)$$

where f_{ij} corresponds to the values in the *i*-th row and *j*-th column of Table 6.3.

Since $c(A \Rightarrow B) + c(A \Rightarrow \overline{B}) = 1$, therefore,

$$\begin{aligned}\Delta &= | c(A \Rightarrow B) - c(A \Rightarrow \overline{B}) | \\ &= | 2c(A \Rightarrow B) - 1 | \\ &\propto c(A \Rightarrow B).\end{aligned}$$

Thus, the standard confidence measure is sufficient to detect an interesting contrast of this type.

Example 2: cr$_2$ (Difference of proportion)

An interesting contrast could be considered between rules $B \Rightarrow A$ and $\overline{B} \Rightarrow A$. An example of this contrast was shown in Figure 6.2. Once again, let confidence be the selected measure for both rules. Let absolute difference be the comparison function. We can summarize this type of contrast as follows:

- *br*: $\{B \Rightarrow A\}$
- υ(*br*): $\{\overline{B} \Rightarrow A\}$
- *M*: <confidence, confidence>
- Δ: absolute difference

The evaluation criterion for this example is shown in Eq. 6.6, where Δ is defined as follows:

$$\begin{aligned}\Delta &= | c(r) - c\ (\upsilon(r)) | \\ &= | c(B \Rightarrow A) - c\ (\overline{B} \Rightarrow A) | \\ &= \left| \frac{f_{11}}{f_{+1}} - \frac{f_{12}}{f_{+2}} \right| = \left| \rho (A \Rightarrow B) - \rho (A \Rightarrow \overline{B}) \right|\end{aligned} \tag{6.6}$$

where ρ, is the rule proportion (Agresti, 2002) and is defined in Eq. 6.7.

$$\rho(A \Rightarrow B) = \frac{P(AB)}{P(B)} = c(B \Rightarrow A) \tag{6.7}$$

Example 3: cr$_3$ (Correlation and Chi-Square)

Correlation is a broadly used statistical measure for analyzing the relationship between two variables. The correlation between *A* and *B* in Table 6.3 is measured as follows:

$$corr = \frac{f_{11} f_{22} - f_{12} f_{21}}{\sqrt{f_{1+} f_{+1} f_{2+} f_{+2}}} \tag{6.8}$$

The correlation measure compares the contrast between the following set of base rules and their neighborhood rules:

- *br* is $\{A \Rightarrow B, B \Rightarrow A, \overline{A} \Rightarrow \overline{B}, \overline{B} \Rightarrow \overline{A}\}$
- υ(*br*) is $\{A \Rightarrow \overline{B}, \overline{B} \Rightarrow A, \overline{A} \Rightarrow B, B \Rightarrow \overline{A}\}$
- *M*: <confidence, confidence>,
- Δ: The difference in the square root of confidence products (see Eq. 6.9).

$$\Delta = \sqrt{c_1 c_2 c_3 c_4} - \sqrt{c_5 c_6 c_7 c_8} \tag{6.9}$$

where c_1, c_2, c_3, c_4, c_5, c_6, c_7, and c_8 correspond to $c(A \Rightarrow B)$, $c(B \Rightarrow A)$, $c(\overline{A} \Rightarrow \overline{B})$, $c(\overline{B} \Rightarrow \overline{A})$, $c(A \Rightarrow \overline{B})$, $c(\overline{B} \Rightarrow A)$, $c(\overline{A} \Rightarrow B)$, and $c(B \Rightarrow \overline{A})$ respectively. Eq. 6.10 is obtained by expanding Eq. 6.9.

$$\Delta = \sqrt{\frac{P(AB)}{P(A)} \frac{P(AB)}{P(B)} \frac{P(\overline{A}\overline{B})}{P(\overline{A})} \frac{P(\overline{A}\overline{B})}{P(\overline{B})}} - \sqrt{\frac{P(A\overline{B})}{P(A)} \frac{P(A\overline{B})}{P(\overline{B})} \frac{P(\overline{A}B)}{P(\overline{A})} \frac{P(\overline{A}B)}{P(B)}} \tag{6.10}$$

$$\Delta = \frac{P(AB)P(\overline{A}\,\overline{B}) - P(A\overline{B})P(\overline{A}B)}{\sqrt{P(A)P(B)P(\overline{A})P(\overline{B})}} \tag{6.11}$$

Eq. 6.11 is the correlation between *A* and *B* as shown in Eq. 8. Chi-square measure is related to correlation in the following way:

$$corr = \sqrt{\chi^2 / N} \tag{6.12}$$

Therefore, both measures are essentially comparing the same type of contrast.

Contrast rules and interestingness measures

Different measures have different perspectives on finding interesting rules. Specifically, each measure defines a base rule and a neighborhood of rules from which interesting contrast rules can be detected. In our proposed approach a user can choose a measure and detect the corresponding contrast rules. In addition, the user has flexibility to choose a base rule/attribute according to what he or she is interested in. Then he or she selects the neighborhood rules as well as the measures to detect the base rule and its neighborhood. This is similar to rule template approaches (see 6.2.1). We implemented examples 1-3 for LON-CAPA data sets, which will be explained in section 6.5.

6.4 Algorithm

In this section we propose an algorithm to find surprising and interesting rules based on the characteristics of different segments of students/problems. The difficulty with algorithms such as Apriori is that when the minimum-support is high, we miss many interesting, but infrequent patterns. On the other hand if we choose a minimum-support that is too low the Apriori algorithm will discover so many rules that finding interesting ones becomes difficult.

Mining Contrast Rules (MCR) Algorithm:

```
Input: $D$ – Input set of $N$ transactions
$B$ – Target variable, the basis of interesting contrasts
$\sigma$ – Minimum (very) low support
$m$ – A measure for ranking the rules
$k$ – Number of the most interesting rules
Divide data set $D$ based on the values of the target variable
foreach $j$ in $B$
    Select $D(j)$, a subset of transactions including $j$
    Find the set of closed frequent itemsets, $L(j)$ within $D(j)$
    foreach $\ell \in L(j)$
        Generate rule $\ell \Rightarrow j$
        Compute measure $m(\ell \Rightarrow j)$
    end
end
Find common rules among the different groups of rules
foreach $br$ and $\upsilon(br)$ pair compute difference in measures, $\Delta$
Sort the rules with respect to $\Delta$
Select top $k$ rules
return $R$
```

Figure 6.7 Mining Contrast Rules (MCR) algorithm for discovering interesting candidate rules

In order to employ the MCR algorithm, several steps must be taken. During the preprocessing phase, we remove items whose support is too high. For example, if 95% of students pass the course, this attribute will be removed from the itemsets so that it does not overwhelm other, more subtle rules. Then we must also select the target variable of the rules to be compared. This allows the user to focus the search space on subjectively interesting rules. If the target variable has C distinct values, we divide the data set, D, into

C disjoint subsets based on the elements of the target variable, as shown in Figure 6.7. For example, in the case where gender is the target variable, we divide the transactions into male and female subsets to permit examination of rule coverage.

Using Borgelt's implementation[13] of the Apriori algorithm (version 4.21), we can find closed itemsets employing a simple filtering approach on the prefix tree (Borgelt, 2003). A closed itemset is a set of items for which none of its supersets have exactly the same support as itself. The advantage of using closed frequent itemsets for our purposes is that we can focus on a smaller number of rules for analysis, and larger frequent itemsets, by discarding the redundant supersets.

We choose a very low minimum support to obtain as many frequent itemsets as is possible. Using perl scripts, we find the common rules between two contrast subsets. Finally, we rank the common rules with all of the previously explained measures, and then the top *k* rules of the sorted ranked-rules are chosen as a candidate set of interesting rules. Therefore an important parameter for this algorithm is minimum support, σ; the lower the σ, the larger the number of common rules. If the user selects a specific ranking measure, *m*, then the algorithm will rank the rules with respect to that measure.

6.5 Experiments

In this section we first provide a general model for data attributes, data sets and their selected attributes, and then explain how we handle continuous attributes. Finally, we discuss our results and experimental issues.

[13] The code for this program is available at http://fuzzy.cs.uni-magdeburg.de/~borgelt/apriori.html.

6.5.1 Data model

In order to better understand the interactions between students and the online educational system, a model is required to analyze the data. Ideally, this model would be both descriptive and predictive in nature.

As shown in Figure 6.8, each student is characterized by a set of attributes which are static for any particular analysis (GPA, gender, ethnicity, etc.) and can be easily quantized. The *u*-tuple ($S_i^{(1)}, S_i^{(2)}$, …, $S_i^{(u)}$) describes the characteristics of the *i*-th student. The set of problems is determined by the scope of the analysis – at this time, single courses over individual terms, but with future possibilities for multi-term analysis – and characterized by a set of attributes, some of which are fixed (Bloom's taxonomic categorization, content type, simulation-dependent, etc.). The *v*-tuple ($P_j^{(1)}, P_j^{(2)}$, …, $P_j^{(v)}$) describes the characteristics of the *j*-th problem.

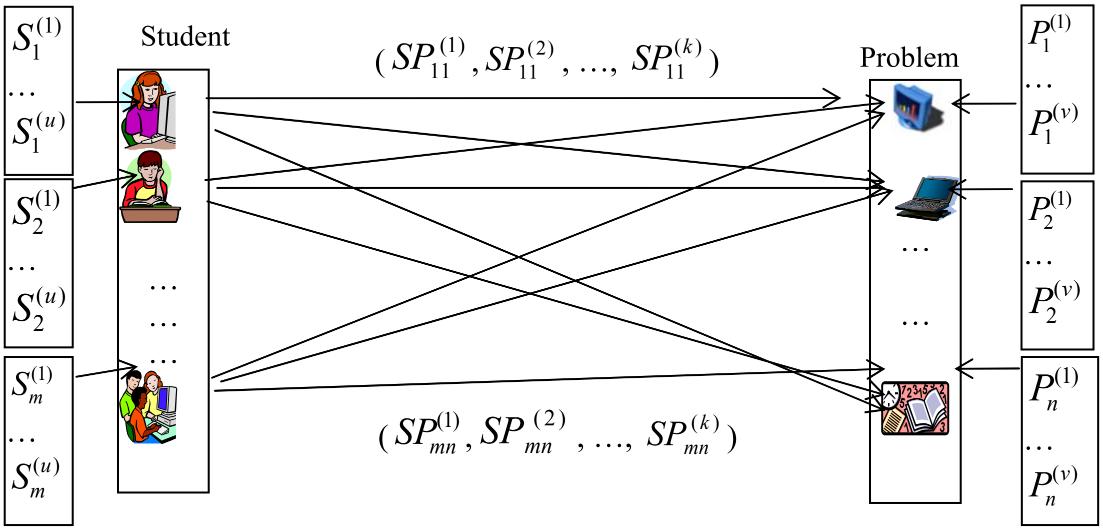


Figure 6.8 Attribute mining model, Fixed students' attributes, Problem attributes, and Linking attributes between students and problem

The interaction of these two sets becomes a third space where larger questions can be asked. The *k*-tuple ($SP_{ij}^{(1)}, SP_{ij}^{(2)}$, …, $SP_{ij}^{(k)}$) describes the characteristics of the *i*-th student linking to the *j*-th problem. LON-CAPA records and dynamically organizes a vast amount of information on students' interactions with and understanding of these materials.

The model is framed around the interactions of the two main sources of interpretable data: students and assessment tasks (problems). Figure 6.9 shows the actual data model, which is frequently called an entity relationship diagram (ERD) since it depicts categories of data in terms of entities and relationships.

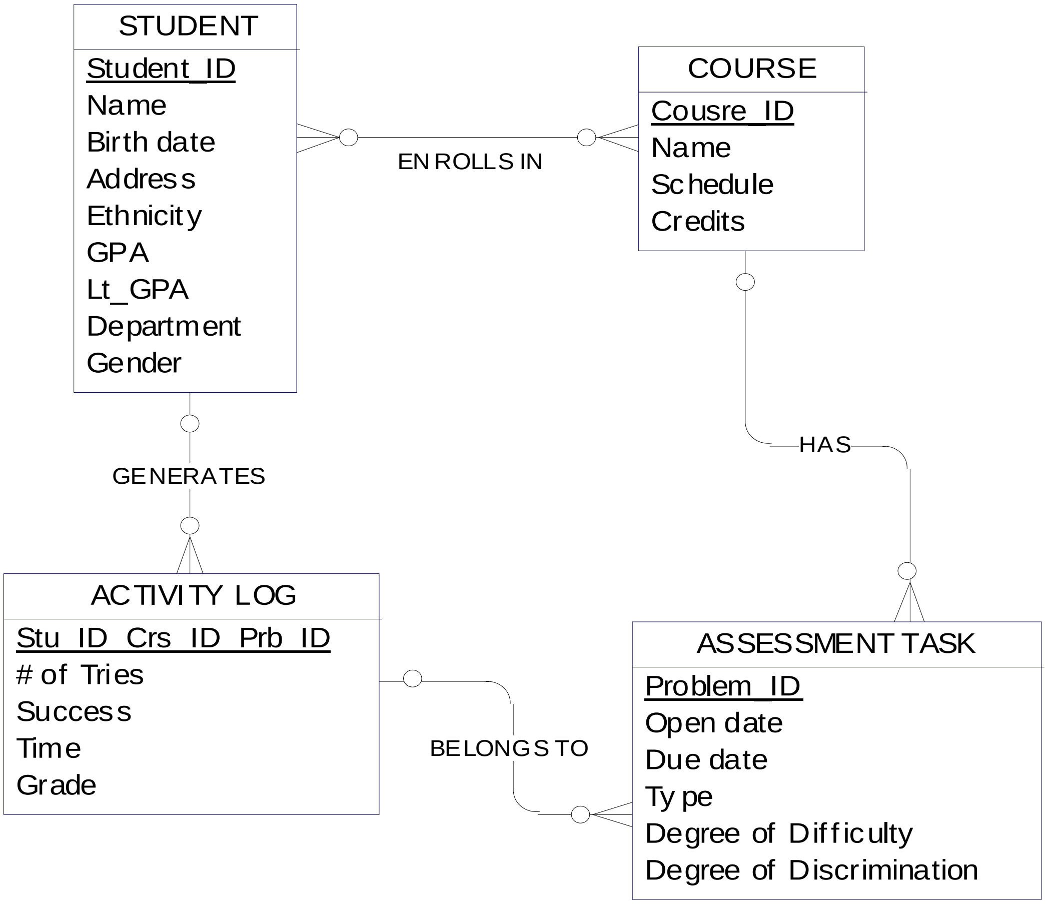


Figure 6.9 Entity Relationship Diagram for a LON-CAPA course

The attributes selected for association analysis are divided into four groups within the LON-CAPA system:

a) Student attributes: which are fixed for any student. Attributes such as Ethnicity, Major, and Age were not included in the data out of necessity – the focus of this work is primarily on the LON-CAPA system itself, so the demographics of students is less relevant. As a result, the following three attributes are included:

GPA: is a continuous variable that is discretized into eight intervals between zero and four with a 0.5 distance.

Gender: is a binary attribute with values Female and Male.

LtGPA (Level Transferred (i.e. High School) GPA)**:** measured the same as GPA

b) Problem attributes: which are fixed for any problem. Among several attributes for the problems we selected the four following attributes:

DoDiff (degree of difficulty)**:** This is a useful factor for an instructor to determine whether a problem has an appropriate level of difficulty. DoDiff is computed by the total number of students' submissions divided by the number of students who solved the problem correctly. Thus, DoDiff is a continuous variable in the interval [0,1] which is discretized into terciles of roughly equal frequency: easy, medium, and hard.

DoDisc (degree of discrimination)**:** A second measure of a problem's usefulness in assessing performance is its discrimination index. It is derived by comparing how students whose performance places them in the top quartile of the class score on that problem compared to those in the bottom quartile. The possible values for DoDisc vary from –1 to +1. A negative value means that students in the lower quartile scored better on that problem than those in the upper. A value close to +1 indicates the higher achieving students (overall) performed better on the problem. We discretize this continuous value into terciles of roughly equal frequency: negatively-discriminating, non-discriminating, and positively-discriminating.

AvgTries (average number of tries)**:** This is a continuous variable which is discretized into terciles of roughly equal frequency: low, medium, and high.

c) Student/Problem interaction attributes: We have extracted the following attributes per student per problem from the activity log:

Succ: Success on the problem (YES, NO)

Tries: Total number of attempts before final answer.

Time: Total time from first attempt until the final answer is derived.

d) Student/Course interaction attributes: We have extracted the following attributes per student per course from the LON-CAPA system.

Grade: Student's Grade, the nine possible labels for grade (a 4.0 scale with 0.5 increments). An aggregation of "grade" attributes is added to the total attribute list.

Pass-Fail: Categorize students with one of two class labels: "Pass" for grades above 2.0, and "Fail" for grades less than or equal to 2.0.

6.5.2 Data sets

For this research we selected three data sets from the LON-CAPA courses as shown in Table 6.4. For example the second row of the table shows that BS111 (Biological Science: Cells and Molecules) integrated 235 online homework problems, and 382 students used LON-CAPA for this course. BS111 had an activity log with approximately 239 MB of data. Though BS111 is a larger course than LBS271 (first row of the table), a physics course, it is much smaller than CEM141 (third row), general chemistry I. This course had 2048 students enrolled and its activity log exceeds 750MB, corresponding to more than 190k student/problem interactions when students attempting to solve homework problems.

Table 6.4 Characteristics of three MSU courses which used LON-CAPA in fall semester 2003

Data set	Course Title	# of Students	# of Problems	Size of Activity log	# of Interactions
LBS 271	Physics_I	200	174	152.1 MB	32,394
BS 111	BiologicalScience	382	235	239.4 MB	71,675
CEM141	Chemistry_I	2048	114	754.8 MB	190,859

For this chapter we focus on two target variables, gender and pass-fail grades, in order to find the contrast rules involving these attributes. A constant difficulty in using any of the association rule mining algorithms is that they can only operate on binary data sets. Thus, in order to analyze quantitative or categorical attributes, some modifications are required – binarization – to partition the values of continuous attributes into discrete intervals and substitute a binary item for each discretized item. In this experiment, we mainly use equal-frequency binning for discretizing the attributes.

6.5.3 Results

This section presents some examples of the interesting contrast rules obtained from the LON-CAPA data sets. Since our approach is an unsupervised case, it requires some practical methods to validate the process. The interestingness of a rule can be subjectively measured in terms of its actionability (usefulness) or its unexpectedness (Silberschatz &

Tuzhilin, 1995; Piatetsky-Shapiro & Matheus, 1994; Liu et al., 1999; Silberschatz & Tuzhilin, 1996).

One of the techniques for mining interesting association rules based on unexpectedness. Therefore, we divide the set of discovered rules into three categories:

1. *Expected and previously known*: This type of rule confirms user beliefs, and can be used to validate our approach. Though perhaps already known, many of these rules are still useful for the user as a form of empirical verification of expectations. For our specific situation (education) this approach provides opportunity for rigorous justification of many long-held beliefs.
2. *Unexpected*: This type of rule contradicts user beliefs. This group of unanticipated correlations can supply interesting rules, yet their interestingness and possible actionability still requires further investigation.
3. *Unknown*: This type of rule does not clearly belong to any category, and should be categorized by domain-specific experts. For our situations, classifying these complicated rules would involve consultation with not only the course instructors and coordinators, but also educational researchers and psychologists.

The following rule tables present five examples of the extracted contrast rules obtained using our approach. Each table shows the coded contrast rule and the “support” and “confidence” of that rule. Abbreviations are used in the rule code, and are summarized as follows: *Succ* stands for success per student per problem, *LtGPA* stands for transfer GPA, *DoDiff* stands for degree of difficulty of a particular problem, and

DoDisc stands for degree of discrimination of a problem. In our experiments, we used three measures to rank the contrast rules:

6.5.3.1 Difference of confidences

The focus of this measure is on comparing the confidences of the contrast rules ($A \Rightarrow B$ and $A \Rightarrow \overline{B}$). Therefore, top rules found by this measure have a high value of confidence ratio (c_1/c_2). Contrast rules in Table 6.5 suggest that students in LBS 271 who are successful in homework problems are more likely to pass the course, and this comes with a confidence ratio c_1/c_2=12.7.

Table 6.5 LBS_271 data set, difference of confidences measure

Contrast Rules	Support & Confidence
(Succ=YES) ==> Passed	(s=86.1%, c=92.7%)
(Succ=YES) ==> Failed	(s=6.8%, c=7.3%)

This rule implies a strong correlation among the student's success in homework problems and his/her final grade. Therefore, this rule belongs to the first category; it is a known, expected rule that validates our approach.

Table 6.6 CEM_141 data set, difference of confidences measure

Contrast Rules	Support & Confidence
(Lt_GPA=[1.5,2)) ==> Passed	(s=0.6%, c=7.7%)
(Lt_GPA=[1.5,2)) ==> Failed	(s=7.1%, c=92.3%)

Contrast rules in Table 6.6 could belong to the first category as well; students with low transfer GPAs are more likely to fail CEM 141 (c_2/c_1=12). This rule has the advantage of actionability; so, when students with low transfer GPAs enroll for the course, the system could be designed to provide them with additional help.

6.5.3.2 Difference of Proportions

The focus of this measure is on comparing the rules ($B \Rightarrow A$ and $\overline{B} \Rightarrow A$). Contrast rules in Table 6.7 suggest that historically strong students that take long periods of time between their first (incorrect) solution attempt and subsequent attempts tend to be female. Though this rule may belong to the second category, there is some empirical evidence that female students have better performances over long periods of time than males (Kashy D.; 2001). We found this interesting contrast rules using the difference of confidences to discover the top significant rules for BS 111. Though the support of the rules is low, that is the result would be of an interesting rule with low-support.

Table 6.7 BS_111 data set, difference of proportion measure

Contrast Rules	Support & Confidence
Male ==> (Lt_GPA=[3.5,4] & Time>20_hours)	(s=0.1%, c=26.3%)
Female ==>(Lt_GPA=[3.5,4] & Time>20_hours)	(s=0.6%, c=89.7%)

6.5.3.3 Chi-square

It is a well-known condition in chi-square testing for contingency tables that cell expected values need to be above 5 to guarantee the veracity of the significance levels obtained (Agresti, 2002). We do pruning if this limitation is violated in some cases, and this usually happens when the expected support corresponding to f_{11} or f_{12} in Table 6.3 is low.

Table 6.8 CEM_141 data set, chi-square measure

Contrast Rules	Support & Confidence
(Lt_GPA=[3,3.5) & Sex=Male & Tries=1) ==> Passed	(s=4.4%, c=82.7%)
(Lt_GPA=[3,3.5) & Sex=Male & Tries=1) ==> Failed	(s=0.9%, c=17.3%)

Contrast rules in Tables 6.8 suggest that students with transfer GPAs in the range of 3.0 to 3.5 that were male and answered homework problems on the first try were more likely to pass the class than to fail it. (c_1/c_2=4.8). This rule could belong to the second category. We found this rule using the chi-square measure for CEM 141.

Table 6.9 LBS_271 data set, difference of confidences measure

Contrast Rules	Support & Confidence
(DoDiff=medium & DoDisc=non_discriminating & Succ=YES & Tries=1) ==> Passed	(s=28.9%, c=94.1%)
(DoDiff=medium & DoDisc=non_discriminating & Succ=YES & Tries=1) ==> Failed	(s=1.8%, c=6.9%)

Contrast rules in Table 6.9 show more complicated rules for LBS 271 using difference of proportion (c_1/c_2=15.9); these rules belong to the third (unknown) category and further consultation with educational experts is necessary to determine whether or not they are interesting.

6.6 Conclusion

LON-CAPA servers are recording students' activities in large logs. We proposed a general formulation of interesting contrast rules and developed an algorithm to discover a set of contrast rules investigating three different statistical measures. This tool can help instructors to design courses more effectively, detect anomalies, inspire and direct further research, and help students use resources more efficiently. An advantage of this developing approach is its broad functionality in many data mining application domains. Specifically, it allows for contrast rule discovery with very low minimum support, therefore permitting the mining of possibly interesting rules that otherwise would go unnoticed.

More measurements tend to permit discovery of higher coverage rules. A combination of measurements should be employed to find out whether this approach for finding more interesting rules can be improved. In this vein, we plan to extend our work to analysis of other possible kinds of contrast rules. This work has been published in (Minaei-Bidgoli et al., 2004g).

Chapter 7 Summary

This dissertation addresses the issues surrounding the use of a data mining framework within a web-based educational system. We introduce the basic concepts of data mining as well as information about current online educational systems, a background on Intelligent Tutoring Systems, and an overview of the LON-CAPA system. A body of literature has emerged, dealing with the different problems involved in data mining for performing classification and clustering upon web-based educational data. This dissertation positions itself to extend data mining research into web-based educational systems – a new and valuable application. Results of data mining tools help students use the online educational resources more efficiently while allowing instructors, problem authors, and course coordinators to design online materials more effectively.

7.1 Summary of the work

This dissertation provides information about the structure of LON-CAPA data, data retrieval processes, and representation of student statistical information including problem and solution strategies. We explain how we provide assessment tools in LON-CAPA on various aspects of teaching and learning. The LON-CAPA system is used for both formative and summative assessment. Feedback from numerous sources has improved the educational materials considerably, a continuous and cyclic task which data mining has the opportunity to impact.

7.1.1 Predicting Student Performance

The first aim of this dissertation is to provide a data mining tool for classifying student characteristics based on features extracted from their logged data. We can use this tool to predict the group to which any individual student will belong with reasonable precision. Eventually, this information will help students use course resources better, based on the usage of that resource by other students in similar groups. Four tree-based (C5.0, CART, Quest, and Cruise) and five non tree-based classifiers (*k*-nearest neighbor, Bayesian, Parzen window and neural network) are used to segregate student data. Using a combination of multiple classifiers leads to a significant accuracy improvement for various LON-CAPA courses. Weighting the features and using a genetic algorithm to minimize the error rate improves the prediction accuracy by at least 10% in all the cases tested.

The successful implementation of student classification to predict their performance, demonstrates the merits of using the LON-CAPA data for pattern recognition in order to predict the students' final grades based on features extracted from their homework data. We design, implement, and evaluate a series of pattern classifiers with various parameters in order to compare their performance in analyzing a real dataset from the LON-CAPA system.

This approach is easily adaptable to different types of courses, different population sizes, and allows for different features to be analyzed. This work represents a rigorous application of known classifiers as a means of analyzing and comparing usage and performance of students who have taken a technical course that was partially/completely administered via the web.

7.1.2 Clustering ensembles

A second objective of this research is to extend previous theoretical work regarding clustering ensembles with the goal of creating an optimal framework for categorizing web-based educational resources. We propose non-adaptive and adaptive resampling schemes for the integration of multiple clusterings (independent and dependent). Experimental results show improved stability and accuracy for clustering structures obtained via bootstrapping, subsampling, and adaptive techniques. This study shows that meaningful consensus partitions for an entire data set of objects can emerge from clusterings of bootstrap (with replacement) and subsamples (without replacement) of small size.

Empirical studies are conducted on several data sets for different consensus functions, number of partitions in the combination and number of clusters in each component. The results demonstrate that there is a trade-off between the number of clusters per component and the number of partitions, and that the sample size of each partition needed in order to perform the combination process converges to an optimal error rate. These improvements offer insights into specific associations within the data sets. The challenging points of using resampling techniques for maintaining the diversity of partitions are discussed. We show that a critical fraction of data exists such that the structure of an entire data set can be perfectly detected. Subsamples of small sizes can reduce computational costs and measurement complexity for many explorative data mining tasks with distributed sources of data. This empirical study also compares the performance of adaptive and non-adaptive clustering ensembles using different consensus functions on a number of data sets. By focusing attention on the data points with the least consistent clustering assignments, one can better approximate the inter-cluster boundaries and improve clustering accuracy and convergence speed as a function of the number of partitions in the ensemble. The comparison of adaptive and non-adaptive approaches is a new avenue for research, and this study helps to pave the way for the useful application of distributed data mining methods.

7.1.3 Interesting association rules

Finally, this dissertation proposes techniques for discovering interesting associations between student attributes, problem attributes, and solution strategies. We develop an algorithm for the discovery of "interesting" association rules within a web-based educational system. The main focus is on mining interesting contrast rules, which are sets of conjunctive rules describing interesting characteristics of different segments within a population. In the context of web-based educational systems, contrast rules help to identify attributes characterizing patterns of performance disparity between various groups of students. This dissertation presents a general formulation of contrast rules as well as a new algorithm for mining interesting contrast rules.

We address the issue of choosing different measures for the discovery of contrast rules. Different measures have different perspectives on finding interesting rules. Specifically, each measure defines a base rule and a neighborhood of rules from which interesting contrast rules can be detected. In our proposed approach a user can choose a measure and detect the corresponding contrast rules. In addition, the user has flexibility to choose a base rule/attribute according to what he or she is interested in. Then the user selects the neighborhood rules as well as the measures to detect the base rule and its neighborhood.

Examining these contrasts can improve the online educational systems for both teachers and students – allowing for more accurate assessment and more effective evaluation of the learning process.

7.2 Future work

There are several promising directions to extend the work presented in this thesis:

1. Develop a tool to find the effects of different types of problems on student achievement. These problems will be classified to find patterns in which students are successful.
2. Develop techniques that apply student information in helping individuals to use resources more efficiently (recommendation system). As an example, the following suggestion might be made by the system: "You are about to start a test. Other students similar to you, who succeeded in this test, have also accessed Section 5 of Chapter 3. You did not. Would you like to access it now before attempting the test?"
3. Find clusters of learners with similar browsing behavior, given students' browsing data and course contents. Though the implications of this clustering are not completely known at this time, it seems a valid question amidst the other solid and useful applications of this work.
4. Identify those students who are at risk of failure, especially in very large classes. This will help the instructor provide appropriate advising in a timely manner.
5. Identify sequences of strategies that students use in solving homework problems. This may help in detecting anomalies in designed problems and assist instructors in developing more effective homework.

Appendix A: Tree Classifiers Output

C5.0

Using C5.0 for classifying the students: This result shows the error rate in each fold in 10-fold cross-validation, and confusion matrix.

In **2-classes** (Passed, Failed)

```
Fold            Rules
----      ----------------
            No      Errors

   0         9       18.2%
   1         9       22.7%
   2        12       27.3%
   3         5       30.4%
   4         8       17.4%
   5         7       21.7%
   6        10       13.0%
   7         8       17.4%
   8         4       17.4%
   9         8       21.7%

  Mean     8.0       20.7%
SE       0.7        1.6%
```

In **3-classes** (High, Middle, Low) we got the following results:

```
Fold        Decision Tree
----      ----------------
          Size      Errors

   0         7       36.4%
   1        12       45.5%
   2         6       45.5%
   3         7       47.8%
   4        10       34.8%
   5         9       34.8%
   6         6       47.8%
   7         8       43.5%
   8        10       47.8%
   9         9       47.8%

  Mean     8.4       43.2%
  SE       0.6        1.8%
```

In **9-classes** we got the following results:

```
Fold        Decision Tree
----      ----------------
          Size      Errors

   0        57       81.8%
   1        51       63.6%
   2        55       63.6%
   3        61       78.3%
   4        48       73.9%
   5        56       73.9%
   6        58       69.6%
   7        53       87.0%
   8        56       78.3%
   9        56       73.9%

  Mean    55.1       74.4%
  SE       1.2        2.4%
```

```
           (a)   (b)   (c)   (d)   (e)   (f)   (g)   (h)   (i)    <-classified as
          ----  ----  ----  ----  ----  ----  ----  ----  ----
                         1           1                              (a): class 1
             1           3     2     2     2                        (b): class 2
             1     2     7     6     2     7     3                  (c): class 3
                   4     2     5     2     4     3     3            (d): class 4
                   2     5     3    12    11     8     2            (e): class 5
                         7     5     9    15     9     7            (f): class 6
                         4     6     3    15     5     8            (g): class 7
                         1     1     7     3     2    14            (h): class 8
                                                                    (i): class 9
```

Here, there are a sample of rule sets resulted the from C5.0 in 3-class classification

```
Rule 1: (8, lift 2.9)
        FirstCorrect > 64
        FirstCorrect <= 112
        TotalCorrect > 181
        AvgTries > 1270
        TotalTimeSpent <= 87.87
        Discussion <= 0
        ->  class High  [0.900]

Rule 2: (5, lift 2.8)
        FirstCorrect > 93
        FirstCorrect <= 99
        TotalCorrect > 181
        AvgTries <= 1270
        Discussion <= 14
        ->  class High  [0.857]

Rule 3: (15/2, lift 2.7)
        FirstCorrect <= 112
        TotalCorrect > 181
        Discussion > 0
        Discussion <= 14
        ->  class High  [0.824]

Rule 4: (8/1, lift 2.6)
        FirstCorrect <= 112
        TotalCorrect > 174
        TotalCorrect <= 180
        AvgTries <= 1768
        Discussion <= 0
        ->  class High  [0.800]

Rule 5: (3, lift 2.6)
        FirstCorrect > 112
        FirstCorrect <= 117
        TotalCorrect > 180
        TotalTimeSpent > 14.01
        Discussion <= 1
        ->  class High  [0.800]
…………

Rule 15: (3/1, lift 2.2)
        FirstCorrect <= 112
        TotalCorrect > 180
        TotalCorrect <= 181
        Discussion <= 0
        ->  class Low  [0.600]
```

Here, there are a sample of rule sets resulted the from C5.0 in 2-class classification

```
Rules:

Rule 1: (158/25, lift 1.2)
        TotalCorrect > 165
        ->  class Passed  [0.838]

Rule 2: (45/8, lift 1.1)
        Discussion > 1
        ->  class Passed  [0.809]

Rule 3: (7, lift 3.2)
        FirstCorrect <= 78
        TotalCorrect <= 165
        ->  class Failed  [0.889]

Rule 4: (2, lift 2.7)
        TotalCorrect <= 165
        AvgTries > 669
        Discussion > 1
        Discussion <= 4
        ->  class Failed  [0.750]

Rule 5: (47/15, lift 2.4)
        TotalCorrect <= 165
        ->  class Failed  [0.673]

Default class: Passed

Evaluation on hold-out data (22 cases):
                Rules
          ----------------
            No      Errors
             5    3(13.6%)   <<
```

And a sample of tree, which is produced by C5.0 in one of the folds in 3 classes:

```
TotalCorrect <= 165:
:...AvgTries > 850: Low (13/2)
:   AvgTries <= 850:
:   :...Discussion > 2:
:       :...TotalTimeSpent <= 20.57: Low (2)
:       :   TotalTimeSpent > 20.57: Middle (3/1)
:       Discussion <= 2:
:       :...TotalTimeSpent > 22.63: Low (8)
:           TotalTimeSpent <= 22.63:
:           :...AvgTries <= 561: Low (7/1)
:               AvgTries > 561:
:               :...TotalCorrect > 156: Low (2)
:                   TotalCorrect <= 156:
:                   :...TotalCorrect <= 136: Low (3/1)
:                       TotalCorrect > 136: Middle (6)
TotalCorrect > 165:
:...AvgTries <= 535:
    :...TotalCorrect <= 177: Low (5)
    :   TotalCorrect > 177: High (5/2)
    AvgTries > 535:
    :...FirstCorrect > 112:
        :...TotalCorrect <= 172: Middle (6)
        :   TotalCorrect > 172:
        :   :...TotalCorrect > 180: Middle (38/13)
        :       TotalCorrect <= 180:
        :       :...TimeTillCorr <= 23.47:
        :           :...TotalCorrect > 178: High (2)
        :           :   TotalCorrect <= 178:
        :           :   :...TotalCorrect <= 174: High (4/2)
        :           :       TotalCorrect > 174: Middle (8/1)
        :           TimeTillCorr > 23.47:
        :           :...FirstCorrect > 129: Middle (2)
        :               FirstCorrect <= 129:
        :               :...TotalCorrect > 175: Low (7/1)
        :                   TotalCorrect <= 175:
        :                   :...FirstCorrect <= 118: Low (2)
        :                       FirstCorrect > 118: High (2)
        FirstCorrect <= 112:
        :...TotalTimeSpent > 87.87: Middle (5/1)
            TotalTimeSpent <= 87.87:
            :...TotalCorrect <= 169: High (5/1)
                TotalCorrect > 169:
                :...TotalCorrect <= 174: Middle (8)
                    TotalCorrect > 174:
                    :...Discussion > 7: Middle (5/1)
                        Discussion <= 7:
                        :...TotalCorrect <= 177: High (5/1)
                            TotalCorrect > 177:
                            :...TotalCorrect <= 181:
                                :...AvgTries <= 1023: High (3)
                                :   AvgTries > 1023: Middle (9/2)
                                TotalCorrect > 181:
                                :...Discussion > 0: High (15/2)
                                    Discussion <= 0:
                                    :...FirstCorrect > 99: Middle (5/1)
                                        FirstCorrect <= 99:
                                        :...FirstCorrect > 89: High (7/1)
                                            FirstCorrect <= 89:
                                            :...AvgTries <= 1355: Middle (4)
                                                AvgTries > 1355: [S1]

Evaluation on hold-out data (22 cases):
            Decision Tree
          ----------------
          Size      Errors
            32    7(31.8%)   <<
```

CART

Some of CART report for 2-Classes using Gini criterion:

File: PHY183.XLS

Target Variable: CLASS2

Predictor Variables: FIRSTCRR, TOTCORR, TRIES, SLVDTIME, TOTTIME, DISCUSS

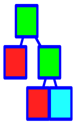
Tree Sequence

Tree Number	Terminal Nodes	Cross-Validated Relative Cost	Resubstitution Relative Cost	Complexity
1	23	0.873 ± 0.099	0.317	-1.000
2	22	0.984 ± 0.104	0.317	1.00E-005
3	15	1.016 ± 0.104	0.397	0.003
4	9	0.762 ± 0.089	0.476	0.004
5	7	0.778 ± 0.091	0.508	0.004
6	5	0.841 ± 0.093	0.556	0.007
7**	3	0.667 ± 0.090	0.619	0.009
8	2	0.714 ± 0.088	0.683	0.018
9	1	1.000 ± 6.73E-005	1.000	0.088

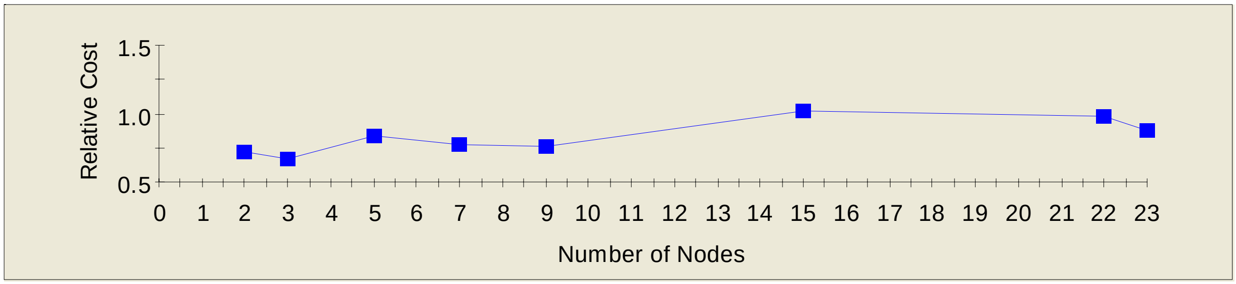
* Minimum Cost

** Optimal

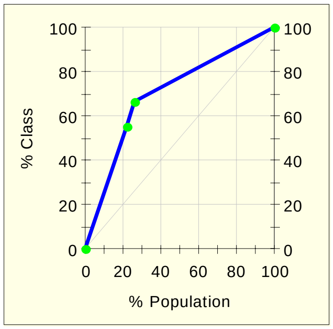
Classification tree topology for: CLASS2



Error Curve



Gains for 2



Gains Data for 2

Node	Cases Class 2	% of Node Class 2	% Class 2	Cum % Class 2	Cum % Pop	% Pop	Cases in Node	Cum lift	Lift Pop
1	35	70.00	55.56	55.56	22.03	22.03	50	2.522	2.522
2	7	70.00	11.11	66.67	26.43	4.41	10	2.522	2.522
3	21	12.57	33.33	100.00	100.00	73.57	167	1.000	0.453

Variable Importance

<table>
<tr><th>Variable</th><th></th><th></th></tr>
<tr><td>TOTCORR</td><td>100.00</td><td>||||||||||||||||||||||||||||||||||||||||||</td></tr>
<tr><td>TRIES</td><td>56.32</td><td>|||||||||||||||||||||||</td></tr>
<tr><td>FIRSTCRR</td><td>4.58</td><td>|</td></tr>
<tr><td>TOTTIME</td><td>0.91</td><td></td></tr>
<tr><td>SLVDTIME</td><td>0.83</td><td></td></tr>
<tr><td>DISCUSS</td><td>0.00</td><td></td></tr>
</table>

Misclassification for Learn Data

Class	N Cases	N Mis-Classed	Pct Error	Cost
1	164	18	10.98	0.11
2	63	21	33.33	0.33

Misclassification for Test Data

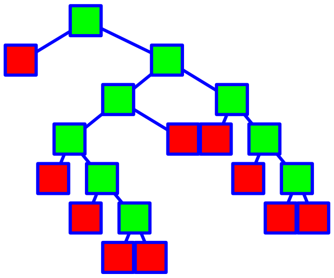
Class	N Cases	N Mis-Classed	Pct Error	Cost
1	164	21	12.80	0.13
2	63	21	33.33	0.33

Some of CART report for 3-Classes using Twoing criterion: (10-fold Cross-Validation):

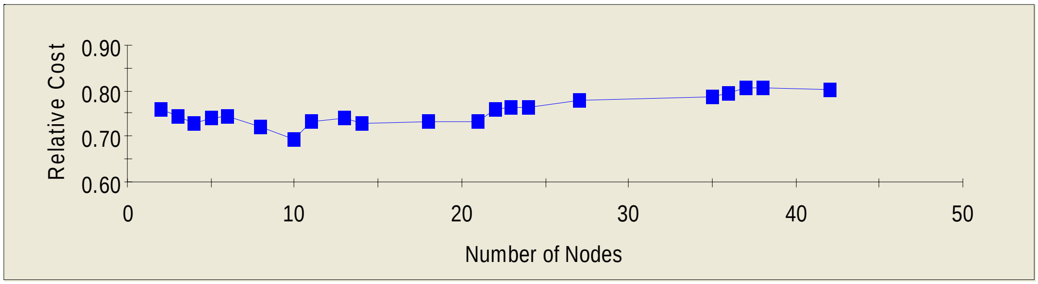
Tree Sequence

Tree Number	Terminal Nodes	Cross-Validated Relative Cost	Resubstitution Relative Cost	Complexity
1	42	0.802 ± 0.050	0.230	-1.000
2	38	0.808 ± 0.050	0.236	0.001
3	37	0.808 ± 0.050	0.238	0.001
4	36	0.794 ± 0.050	0.242	0.003
5	35	0.786 ± 0.050	0.247	0.003
6	27	0.778 ± 0.050	0.289	0.004
7	24	0.762 ± 0.050	0.311	0.005
8	23	0.762 ± 0.050	0.319	0.005
9	22	0.761 ± 0.050	0.327	0.005
10	21	0.731 ± 0.049	0.336	0.006
11	18	0.734 ± 0.049	0.366	0.007
12	14	0.727 ± 0.049	0.407	0.007
13	13	0.740 ± 0.049	0.418	0.007
14	11	0.732 ± 0.049	0.444	0.009
15**	10	0.694 ± 0.049	0.457	0.009
16	8	0.720 ± 0.050	0.500	0.014
17	6	0.743 ± 0.050	0.545	0.015
18	5	0.741 ± 0.050	0.574	0.019
19	4	0.728 ± 0.050	0.605	0.021
20	3	0.745 ± 0.050	0.661	0.037
21	2	0.758 ± 0.035	0.751	0.060
22	1	1.000 ± 0.000	1.000	0.166

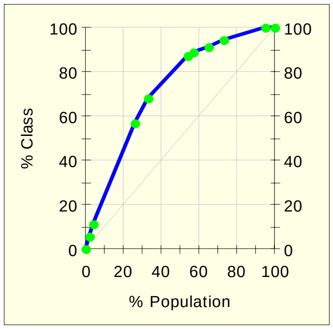
Classification tree topology for: CLASS3



Error Curve



Gains for 1



Gains Data for 1

Node	Cases Class 1	% of Node Class 1	% Class 1	Cum % Class 1	Cum % Pop	% Pop	Cases in Node	Cum lift	Lift Pop
9	4	80.00	5.80	5.80	2.20	2.20	5	2.632	2.632
2	4	80.00	5.80	11.59	4.41	2.20	5	2.632	2.632
6	31	62.00	44.93	56.52	26.43	22.03	50	2.138	2.040
4	8	57.14	11.59	68.12	32.60	6.17	14	2.090	1.880
7	13	27.08	18.84	86.96	53.74	21.15	48	1.618	0.891
5	1	12.50	1.45	88.41	57.27	3.52	8	1.544	0.411
10	2	11.76	2.90	91.30	64.76	7.49	17	1.410	0.387
8	2	11.11	2.90	94.20	72.69	7.93	18	1.296	0.366
1	4	8.00	5.80	100.00	94.71	22.03	50	1.056	0.263
3	0	0.00	0.00	100.00	100.00	5.29	12	1.000	0.000

Variable Importance

Variable		
TOTCORR	100.00	||||||||||||||||||||||||||||||||||||||||||
TRIES	40.11	||||||||||||||||
FIRSTCRR	24.44	||||||||||
TOTTIME	23.22	|||||||||
SLVDTIME	21.67	||||||||
DISCUSS	14.44	|||||

Misclassification for Learn Data

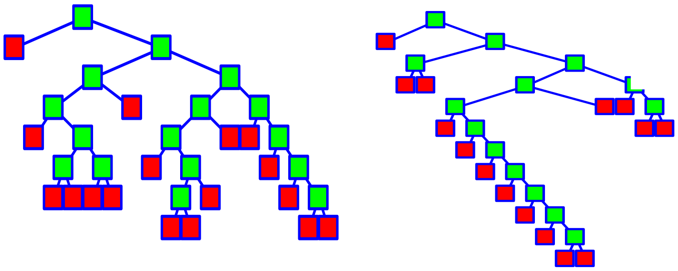
Class	N Cases	N Mis-Classed	Pct Error	Cost
1	69	22	31.88	0.32
2	95	34	35.79	0.36
3	63	15	23.81	0.24

Misclassification for Test Data

Class	N Cases	N Mis-Classed	Pct Error	Cost
1	69	35	50.72	0.51
2	95	52	54.74	0.55
3	63	21	33.33	0.33

Some of CART report for 9-Classes using Entropy criterion: (10-fold Cross-Validation)

Different tree topologies for: CLASS-9



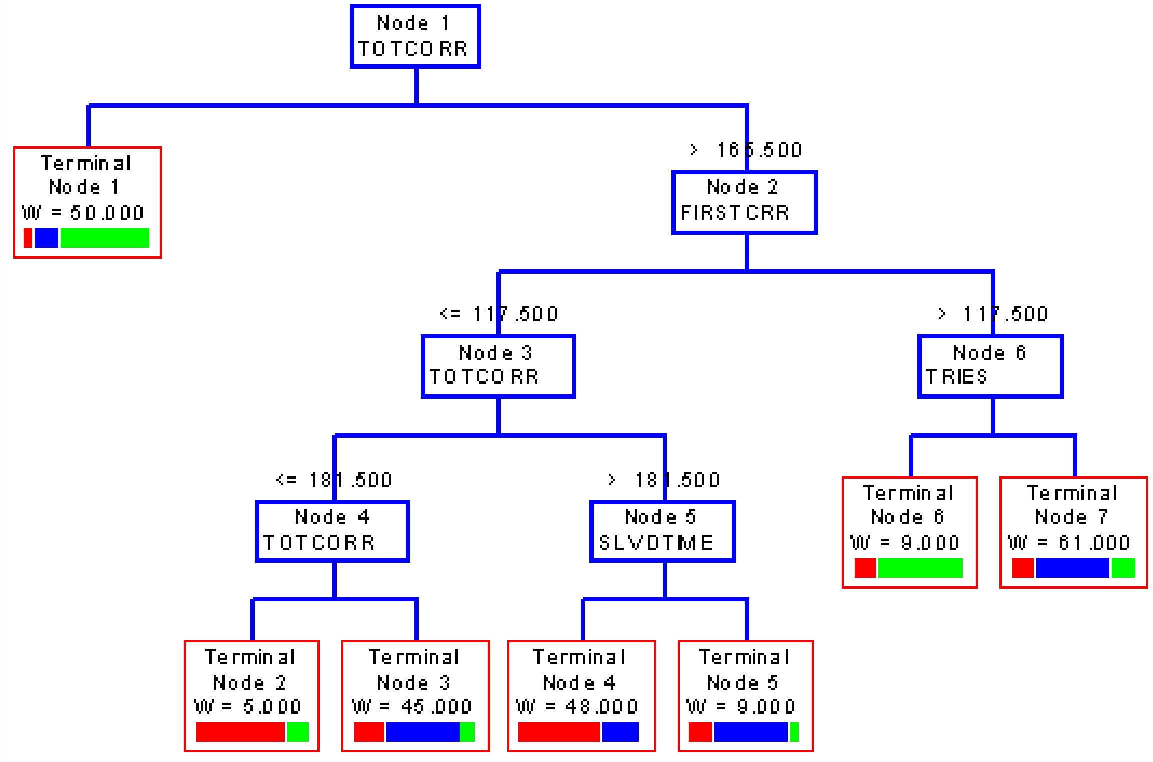
Entropy **Gini** **Twoing**

Descriptive Statistics in CART for 3-Classes

```
   Variable                N         Mean            SD          Min           Max
Sum
 -----------------------------------------------------------------------------------------
---
 Overall
   FIRSTCRR           227.00      106.242        20.462       47.000       150.000
24117.000
   TOTCORR            227.00      171.678        18.155       80.000       184.000
38971.000
   TRIES              227.00      977.987       450.898      265.000      3095.000
222003.000
   SLVDTIME           227.00       36.620        24.837        2.590       130.870
8312.700
   TOTTIME            227.00       37.948        25.434        3.000       130.870
8614.170
   DISCUSS            227.00        1.330         3.034        0.000        23.000
302.000

 CLASS3 = 1
   FIRSTCRR            69.00      103.145        19.598       57.000       149.000
7117.000
   TOTCORR             69.00      179.290         7.900      141.000       184.000
12371.000
   TRIES               69.00     1088.406       439.742      487.000      2227.000
75100.000
   SLVDTIME            69.00       39.060        22.209        2.590        98.840
2695.130
   TOTTIME             69.00       39.764        22.797        3.000        99.200
2743.720
   DISCUSS             69.00        1.493         2.988        0.000        14.000
103.000

 CLASS3 = 2
   FIRSTCRR            95.00      108.505        20.973       54.000       150.000
10308.000
   TOTCORR             95.00      175.453        12.412      118.000       184.000
16668.000
   TRIES               95.00      984.937       443.874      392.000      3095.000
93569.000
   SLVDTIME            95.00       36.866        27.353        4.100       130.870
3502.240
   TOTTIME             95.00       37.916        27.865        4.130       130.870
3602.010
   DISCUSS             95.00        1.537         3.596        0.000        23.000
146.000

 CLASS3 = 3
```

```
   FIRSTCRR             63.00      106.222       20.482       47.000      147.000
6692.000
   TOTCORR              63.00      157.651       24.763       80.000      184.000
9932.000
   TRIES                63.00      846.571      446.210      265.000     2623.000
53334.000
   SLVDTIME             63.00       33.577       23.605        4.870      107.100
2115.330
   TOTTIME              63.00       36.007       24.561        5.920      114.210
2268.440
   DISCUSS              63.00        0.841        1.953        0.000        9.000
53.000
```

A Sample of CART tree for 3-Classes using Entropy criterion: (10-fold Cross-validation)



QUEST

```
Summary of numerical variable: FirstCorr

     Size        Obs        Min        Max       Mean         Sd

      226        226  0.470E+02  0.150E+03  0.106E+03  0.204E+02


Summary of numerical variable: TotCorr

     Size        Obs        Min        Max       Mean         Sd

      226        226  0.960E+02  0.184E+03  0.172E+03  0.171E+02


Summary of numerical variable: AvgTries

     Size        Obs        Min        Max       Mean         Sd

      226        226  0.193E+01  0.169E+02  0.551E+01  0.246E+01


Summary of numerical variable: TimeCorr

     Size        Obs        Min        Max       Mean         Sd

      226        226  0.249E+01  0.942E+02  0.280E+02  0.185E+02


Summary of numerical variable: TimeSpent

     Size        Obs        Min        Max       Mean         Sd

      226        226  0.260E+01  0.942E+02  0.281E+02  0.185E+02


Summary of numerical variable: Discuss

     Size        Obs        Min        Max       Mean         Sd

      226        226  0.000E+00  0.140E+02  0.912E+00  0.201E+01
```

Result for 2-classes

```
  Summary of response variable: Class2
               class   frequency
              Failed      62
              Passed     164
                  -------------
                   2     226

  Options for tree construction
  Learning sample
  estimated priors are
               Class     prior
              Failed    0.27434
              Passed    0.72566
  Size and CV misclassification cost and SE of subtrees:
  Tree   #Tnodes     Mean       SE(Mean)
    1       26      0.1947     0.2634E-01
    2*      16      0.1903     0.2611E-01
    3       13      0.1947     0.2634E-01
    4       12      0.2035     0.2678E-01
    5        6      0.1947     0.2634E-01
    6**      4      0.1947     0.2634E-01
    7        3      0.2301     0.2800E-01
    8        2      0.2434     0.2854E-01
    9        1      0.2743     0.2968E-01

CART 0-SE tree is marked with *
CART SE-rule using CART SE is marked with **

use 10-fold CV sample pruning
  SE-rule trees based on number of SEs = 1.00

  subtree  # Terminal  complexity  current
  number    nodes       value        cost
     1        26       0.0000     0.0885
     2        16       0.0022     0.1106
     3        13       0.0029     0.1195
     4        12       0.0044     0.1239
     5         6       0.0052     0.1549
     6         4       0.0111     0.1770
     7         3       0.0177     0.1947
     8         2       0.0354     0.2301
     9         1       0.0442     0.2743


Classification tree:

      Node 1: TotCorr <= 156.9
        Node 2: Failed
      Node 1: TotCorr > 156.9
        Node 3: TotCorr <= 168.8
          Node 18: Discuss <= 1.279
            Node 20: Failed
          Node 18: Discuss > 1.279
            Node 21: Passed
        Node 3: TotCorr > 168.8
          Node 19: Passed




  Classification matrix based on learning sample
              predicted class
  actual class   Failed   Passed
        Failed       35       27
        Passed       13      151

Classification matrix based on 10-fold CV
              predicted class
  actual class   Failed   Passed
```

```
      Failed       33       29
      Passed       15      149
```

Result for 3-classes

```
  use 10-fold CV sample pruning
  SE-rule trees based on number of SEs = 1.00

  Size and CV misclassification cost and SE of subtrees:
  Tree   #Tnodes     Mean       SE(Mean)
    1       47      0.5354     0.3318E-01
    2       30      0.5265     0.3321E-01
    3       24      0.5354     0.3318E-01
    4       10      0.4735     0.3321E-01
    5*       9      0.4425     0.3304E-01
    6        8      0.4513     0.3310E-01
    7        6      0.4602     0.3315E-01
    8**      4      0.4735     0.3321E-01
    9        2      0.4823     0.3324E-01
   10        1      0.5796     0.3283E-01

  CART 0-SE tree is marked with *
  CART SE-rule using CART SE is marked with **


  Following tree is based on **

  Structure of final tree

  Node Left node Right node   Split variable   Predicted class
    1        2         3        TotCorr
    2  * terminal node *                         Low
    3       26        27        1stGotCrr
   26       28        29        TotCorr
   28  * terminal node *                         Middle
   29  * terminal node *                         High
   27  * terminal node *                         Middle

  Number of terminal nodes of final tree = 4
  Total number of nodes of final tree = 7

  Classification tree:

      Node 1: TotCorr <= 165.5
        Node 2: Low
      Node 1: TotCorr > 165.5
        Node 3: 1stGotCrr <= 117.5
          Node 26: TotCorr <= 181.5
            Node 28: Middle
          Node 26: TotCorr > 181.5
            Node 29: High
        Node 3: 1stGotCrr > 117.5
          Node 27: Middle

  Classification matrix based on learning sample
              predicted class
  actual class     High      Low   Middle
          High       34        4       31
           Low        2       34       26
        Middle       21       11       63

  Classification matrix based on 10-fold CV
              predicted class
  actual class     High      Low   Middle
          High       27       10       32
           Low        4       37       21
        Middle       25       15       55
```

Bagging (Leave-one-out method)

```
  estimated priors are
               Class     prior
```

```
                High    0.30531
                 Low    0.27434
              Middle    0.42035
  minimal node size: 2
  use univariate split
  use (biased) exhaustive search for variable and split selections
  use the divergence famliy
  with lambda value:   0.5000000


  use 226-fold CV sample pruning
  SE-rule trees based on number of SEs = 1.00

  Size and CV misclassification cost and SE of subtrees:
  Tree   #Tnodes     Mean       SE(Mean)
    1       67      0.5354     0.3318E-01
    2       61      0.5354     0.3318E-01
    3       31      0.5177     0.3324E-01
    4       24      0.5000     0.3326E-01
    5*      10      0.4204     0.3283E-01
    6        9      0.4425     0.3304E-01
    7**      8      0.4469     0.3307E-01
    8        6      0.5000     0.3326E-01
    9        4      0.4956     0.3326E-01
   10        2      0.4823     0.3324E-01
   11        1      0.5796     0.3283E-01

  CART 0-SE tree is marked with *
  CART SE-rule using CART SE is marked with **

  Following tree is based on **

  Structure of final tree


  Node Left node Right node   Split variable   Predicted class
    1        2         3        TotCorr
    2  * terminal node *                         Low
    3       32        33        1stGotCrr
   32       34        35        TotCorr
   34       36        37        TotCorr
   36  * terminal node *                         High
   37  * terminal node *                         Middle
   35       70        71        TimeCorr
   70  * terminal node *                         High
   71  * terminal node *                         Middle
   33      104       105        TimeCorr
  104  * terminal node *                         Middle
  105      132       133        TotCorr
  132  * terminal node *                         Low
  133  * terminal node *                         Middle

  Number of terminal nodes of final tree = 8
  Total number of nodes of final tree = 15

Number of terminal nodes of final tree = 10
Total number of nodes of final tree = 19

  Classification tree:

      Node 1: TotCorr <= 165.5
        Node 2: Low
      Node 1: TotCorr > 165.5
        Node 3: 1stGotCrr <= 117.5
          Node 26: TotCorr <= 181.5
            Node 28: TotCorr <= 169.5
              Node 30: High
            Node 28: TotCorr > 169.5
              Node 31: Middle
          Node 26: TotCorr > 181.5
            Node 29: TimeCorr <= 52.44
```

```
              Node 56: High
            Node 29: TimeCorr > 52.44
              Node 57: Middle
        Node 3: 1stGotCrr > 117.5
          Node 27: TimeCorr <= 24.49
            Node 80: Middle
          Node 27: TimeCorr > 24.49
            Node 81: TotCorr <= 180.5
              Node 104: Low
            Node 81: TotCorr > 180.5
              Node 105: 1stGotCrr <= 130.0
                Node 110: TimeCorr <= 27.16
                  Node 112: Low
                Node 110: TimeCorr > 27.16
                  Node 113: Middle
              Node 105: 1stGotCrr > 130.0
                Node 111: High


Classification matrix based on learning sample
              predicted class
  actual class     High      Low   Middle
          High       38        5       26
           Low        3       47       12
        Middle       13       13       69


Classification matrix based on 226-fold CV
              predicted class
  actual class     High      Low   Middle
          High       31        9       29
           Low        5       43       14
        Middle       19       19       57
```

Result for 9-classes

```
   Classification tree:

      Node 1: TotCorr <= 104.0
        Node 2: 0
      Node 1: TotCorr > 104.0
        Node 3: TotCorr <= 165.5
          Node 4: 3
        Node 3: TotCorr > 165.5
          Node 5: TotCorr <= 181.5
            Node 44: TimeCorr <= 2.565
              Node 46: 8
            Node 44: TimeCorr > 2.565
              Node 47: TimeCorr <= 71.06
                Node 48: AvgTries <= 3.145
                  Node 50: 4
                Node 48: AvgTries > 3.145
                  Node 51: 1stGotCrr <= 77.00
                    Node 56: 5
                  Node 51: 1stGotCrr > 77.00
                    Node 57: AvgTries <= 12.52
                      Node 58: TimeCorr <= 40.50
                        Node 60: 5
                      Node 58: TimeCorr > 40.50
                        Node 61: 6
                    Node 57: AvgTries > 12.52
                      Node 59: 3
              Node 47: TimeCorr > 71.06
                Node 49: 3
          Node 5: TotCorr > 181.5
            Node 45: AvgTries <= 2.630
              Node 108: 4
            Node 45: AvgTries > 2.630
              Node 109: 1stGotCrr <= 111.5
                Node 110: 1stGotCrr <= 55.50
                  Node 112: 5
                Node 110: 1stGotCrr > 55.50
                  Node 113: TimeCorr <= 57.44
                    Node 114: AvgTries <= 5.330
                      Node 116: 6
                    Node 114: AvgTries > 5.330
                      Node 117: 8
                  Node 113: TimeCorr > 57.44
                    Node 115: 6
              Node 109: 1stGotCrr > 111.5
                Node 111: 6

          predicted class
actual class        0        2        3        4        5        6        7        8
           0        1        0        0        0        0        0        0        0
           2        1        0        8        0        0        1        0        0
           3        0        0       21        2        2        3        0        0
           4        0        0        7        5        8        2        0        1
           5        0        0        6        0       32        3        0        2
           6        0        0        5        0       19       24        0        4
           7        0        0        2        1       16       15        0        7
           8        0        0        2        0        3        4        0       19
elapsed time: 386.25 seconds  (user: 386.14, system: 0.11)
```

CRUISE

Here, some output results of CRUISE for 3-classes: CV misclassification cost and SE of subtrees:

```
  Subtree      CV R(t)          CV SE       # Terminal Nodes
 (largest)    0.549823        0.3313E-01          82
      1       0.539823        0.3315E-01          70
      2       0.544248        0.3313E-01          67
      3       0.539823        0.3315E-01          59
      4       0.526549        0.3321E-01          56
      5       0.544248        0.3313E-01          41
      6       0.553097        0.3307E-01          38
      7       0.553097        0.3307E-01          23
      8       0.561947        0.3300E-01          21
      9       0.557522        0.3304E-01          15
     10       0.535398        0.3318E-01           9
     11       0.504425        0.3326E-01           8
     12*      0.460177        0.3315E-01           6
     13       0.504425        0.3326E-01           2
     14       0.579646        0.3283E-01           1
```

* denotes 0-SE Tree
** denotes given-SE Tree
* tree is same as ** tree

```
  Following tree is based on **

  Splits of the Tree:

   Node      Split variable
     1  TotCorr
       2  * terminal *
       3  TotCorr
         8  TotCorr
           24  * terminal *
           25  * terminal *
         9  TotCorr
           27  * terminal *
           28  TimeCorr
             84  * terminal *
             85  * terminal *
```

Tree Structure:

```
 Node 1: TotCorr <=  163.156
  Node 2: Terminal Node, predicted class = Low
     Class label :   High    Low Middle
     Class size  :      3     28     11

Node 1: TotCorr >  163.156
  Node 3: TotCorr <=  171.059
    Node 8: TotCorr <=  168.639
      Node 24: Terminal Node, predicted class = Low
         Class label :   High    Low Middle
         Class size  :      2      7      1

    Node 8: TotCorr >  168.639
      Node 25: Terminal Node, predicted class = Middle
         Class label :   High    Low Middle
         Class size  :      3      4      9

  Node 3: TotCorr >  171.059
    Node 9: TotCorr <=  183.206
      Node 27: Terminal Node, predicted class = Middle
         Class label :   High    Low Middle
         Class size  :     34     18     53

    Node 9: TotCorr >  183.206
      Node 28: ABS(TimeCorr -  35.4849    ) <=  19.1162
        Node 84: Terminal Node, predicted class = High
           Class label :   High    Low Middle
           Class size  :     22      4     11

      Node 28: ABS(TimeCorr -  35.4849    ) >  19.1162
        Node 85: Terminal Node, predicted class = Middle
           Class label :   High    Low Middle
           Class size  :      5      1     10
```

Detailed Description of the Tree:

```
Nodes      No.     Subnode    Split     Split          Split
label    cases      label     stat.    variable        value
    1     226          2        F      TotCorr      <=   163.16
                       3                            <   infinity
                   # obs   mean/mode of TotCorr
         Class   High :  69      179.290
         Class    Low :  62      158.903
         Class Middle :  95      175.453

    2      42        **** terminal, predicted class: Low
                      # obs
         Class   High :   3
         Class    Low :  28
         Class Middle :  11

    3     184          8        F      TotCorr      <=   171.06
                       9                            <   infinity
                   # obs   mean/mode of TotCorr
         Class   High :  66      180.576
         Class    Low :  34      175.176
         Class Middle :  84      179.226

    8      26         24        F      TotCorr      <=   168.64
                      25                            <   infinity
                   # obs   mean/mode of TotCorr
         Class   High :   5      167.600
         Class    Low :  11      167.182
         Class Middle :  10      169.800

   24      10        **** terminal, predicted class: Low
                      # obs
         Class   High :   2
```

```
           Class    Low :   7
           Class Middle :   1

     25      16        **** terminal, predicted class: Middle
                        # obs
           Class   High :   3
           Class    Low :   4
           Class Middle :   9

      9     158         27        F      TotCorr      <=   183.21
                        28                            <   infinity
                     # obs   mean/mode of TotCorr
           Class   High :  61      181.639
           Class    Low :  23      179.000
           Class Middle :  74      180.500

     27     105        **** terminal, predicted class: Middle
                        # obs
           Class   High :  34
           Class    Low :  18
           Class Middle :  53

     28      53         84  Levene   ABS(TimeCorr - 35.5    ) <=19.116
                        85                            <   infinity
                     # obs   mean/mode of TimeCorr
           Class   High :  27      34.7652
           Class    Low :   5      36.5700
           Class Middle :  21      36.1519

     84      37        **** terminal, predicted class: High
                        # obs
           Class   High :  22
           Class    Low :   4
           Class Middle :  11

     85      16        **** terminal, predicted class: Middle
                        # obs
           Class   High :   5
           Class    Low :   1
           Class Middle :  10


Number of nodes in maximum tree        =     153
Number of nodes in final tree          =      11
Number of terminal nodes in final tree =       6

 Classification Matrix :         Predicted class
                               High    Low Middle
 Actual class #obs   Prior ---------------------
         High   69   0.305       22      5     42
          Low   62   0.274        4     35     23
       Middle   95   0.420       11     12     72

 Total obs = 226,    # correct = 129
 Resubstitution misclassification cost =   0.4292
 S.E. of resubstitution misclassification cost =   0.3055E-01

 Cross-validation error cost from pruning =   0.4602
 S.E. of CV misclassification cost =   0.3315E-01
 Elapsed system time in seconds:  5.54
```

Bibliography

[1] Aeberhard, S., Coomans D., and de Vel, O. (1992) “Comparison of Classifiers in High Dimensional Settings”, Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland.

[2] Agrawal, R., Imielinski, T.; Swami A. (1993), "Mining Associations between Sets of Items in Massive Databases", *Proc. of the ACM-SIGMOD 1993 Int'l Conference on Management of Data*, Washington D.C., May 1993.

[3] Agrawal, R.; Srikant, R. (1994) "Fast Algorithms for Mining Association Rules", *Proceeding of the 20th International Conference on Very Large Databases*, Santiago, Chile, September 1994.

[4] Agrawal, R. and Srikant. R. (1995) “Mining Sequential Patterns”. In *Proceeding of the 11th International Conference on Data Engineering*, Taipei, Taiwan, March 1995.

[5] Agrawal, R., Shafer, J.C. (1996) "Parallel Mining of Association Rules", IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, December 1996.

[6] Agresti, A. (2002), *Categorical data analysis.* 2nd edition, New York: Wiley, 2002.

[7] Albertelli, G., Minaei-Bigdoli, B., Punch, W.F., Kortemeyer, G., and Kashy, E., (2002) “Concept Feedback In Computer-Assisted Assignments”, *Proceedings of the (IEEE/ASEE) Frontiers in Education conference,* 2002

[8] Albertelli, G. Sakharuk, A., Kortemeyer, G., Kashy, E., (2003) “Individualized examinations for large on-campus courses”, *Proceedings of the (IEEE/ASEE) Frontiers in Education Conference 2003, vol 33.*

[9] Arning, A., Agrawal, R., and Raghavan, P. (1996) “A Linear Method for Deviation Detection in Large Databases” *Proceeding of 2nd international conference on Knowledge Discovery and Data Mining, KDD 1996.*

[10] Azevedo, R, Bernard, R, M, "*A Meta-analysis of the Effects of Feedback in Computer-based Instruction*", J. Educational Computing Research 13, 111-127. (1995).

[11] Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm, *Proceeding ICGA 2*, pp. 14-21, Lawrence Erlbuam Associates, Publishers, 1987.

[12] Baker, E. (1978). *Cluster Analysis by Optimal Decomposition of Induced Fuzzy Sets*. Delft University Press, Delft, The Netherlands.

[13] Baker, E., and Jain, A. K. (1981). "A Clustering performance measure based on fuzzy set decomposition." *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI 3*, 66-75.

[14] Bala J., De Jong K., Huang J., Vafaie H., and Wechsler H. (1997). Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation 4(3) - Special Issue on Evolution, Learning, and Instinct: 100 years of the Baldwin Effect.* 1997.

[15] Bandyopadhyay, S., and Muthy, C.A. (1995). "Pattern Classification Using Genetic Algorithms", *Pattern Recognition Letters*, Vol. 16, pp. 801-808.

[16] Barchman R. J., and Anand T. (1996), *The process of Knowledge Discovery in Database*, 37-57, AAAI/MIT Press.

[17] Barthelemy J.-P. and Leclerc, B. (1995) "The median procedure for partition", In *Partitioning Data Sets, AMS DIMACS Series in Discrete Mathematics*, I.J. Cox et al eds., 19, pp. 3-34, 1995.

[18] Bay, S. D. and Pazzani, M. J., (2001) "Detecting Group Differences: Mining Contrast Sets. *Data Mining and Knowledge Discovery*, 2001, Vol 5, No 3 213-246.

[19] Ben-Hur, A., Elisseeff, A., and Guyon, I. (2002) "*A stability based method for discovering structure in clustered data*," in Pac. Symp. Biocomputing, 2002, vol. 7, pp. 6-17.

[20] Bloom, B. S. (1956). *Taxonomy of Educational Objectives*. New York, NY, McKay.

[21] Bloom, B. S. (1984). "*The 2-Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-one Tutoring*." Educational Researcher 13: 4-16.

[22] Bonham, S, W, Beichner, R, J, and Deardorff, D. L, (2001). "*Online Homework: Does It Make a Difference?*" The Physics Teacher.

[23] Borgelt, C. (2003), "Efficient Implementations of Apriori and Eclat**",** *Workshop of Frequent Item Set Mining Implementations* (FIMI) 2003.

[24] Bransford, J. D., A. L. Brown, et al. (2000). *How People Learn*. Washington, D.C., National Academy Press.

[25] Breiman, L. (1998) "Arcing classifiers", *The Annals of Statistics*, 26(3), 801-849,1998.

[26] Breiman, L., (1996) "Bagging Predictors", *Journal of Machine Learning*, Vol 24, no. 2, 1996, pp 123-140.

[27] Breiman, L., Freidman, J.H., Olshen, R. A., and Stone, P. J. (1984). *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group.

[28] Bruggemann-Klein, A. and Wood, D. (1988). Drawing Trees Nicely with TEX.

[29] Chan, K.C.C. Ching, J.Y. and Wong, A.K.C. (1992). "A Probabilistic Inductive Learning Approach to the Acquisition of Knowledge in Medical Expert Systems". *Proceeding of 5th IEEE Computer Based Medical Systems Symposium*. Durham NC.

[30] CAPA, See *http://capa.msu.edu/*

[31] Cestnik, B. Konenenko, I. Bratko, I. (1987). *ASSISTANT 86: A Knowledge Elicitation Tool for Sophisticated Users*, in Bratko, I. and Navrac, N. (eds), Progress in Machine Learning, Sigma Press, UK.

[32] CLEAR Software, Inc. (1996). *allCLEAR User's Guid*e, CLEAR Software, Inc, 199 Wells Avenue, Newton, MA.

[33] *Chernoff, H. (1973), "The use of Faces to represent Points in k-Dimensional Space Graphically",* Journal of American Statistical Association*, Vol. 68, page 361-368, June 1973.*

[34] Ching, J.Y. Wong, A.K.C. and Chan, C.C. (1995). "Class Dependent Discretisation for Inductive Learning from Continuous and Mixed-mode Data". *IEEE Transaction. PAMI*, 17(7) 641 - 645.

[35] Comon, P. (1994) "Indepenent Component Analysis – a New Concept?" *Signal Processing*, Vol. 36, No. 3, page 287-314.

[36] CourseInfo ^{TM}, Blackboard Inc., Washington, D.C. (CourseInfo is a trademark of Blackboard, Inc., see *http://product.blackboard.net/courseinfo/; http://www.blackboard.com*).

[37] Dan, S.; and Colla. P., (1998) CART--Classification and Regression Trees. San Diego, CA: Salford Systems, 1998.

[38] De Jong K.A., Spears W.M. and Gordon D.F. (1993). Using genetic algorithms for concept learning. *Machine Learning* 13, 161-188, 1993.

[39] Dempster, A.P., Laird, N. M., and Rubin, D. B. (1977) *Maximum Likelihood From Incomplete Data Via the EM Algorithm*, Journal of the Royal Statistical Society B, 39: 1-22, 1977.

[40] Ding, Q., Canton, M., Diaz, D., Zou, Q., Lu, B., Roy, A., Zhou, J., Wei, Q., Habib, A., Khan, M.A. (2000), "Data *mining* survey", Computer Science Dept., North Dakota State University. See *http://midas.cs.ndsu.nodak.edu/~ding/datamining/dm-survey.doc*

[41] Dixon, J.K. (1979) "Pattern recognition with partly missing data" *IEEE Transaction on Systems, Man and Cybernetics SMC* 9, 617-621

[42] Dhillon, I. S. and Modha, D. S., (2000)"A Data-clustering Algorithm on Distributed Memory Multiprocessors", In Proceedings of *Large-scale Parallel KDD Systems Workshop, ACM SIGKDD, in Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence*, 2000, 1759: 245-260.

[43] Dong, G., Li, J., (1998) "Interestingness of Discovered Association Rules in terms of Neighborhood-Based Unexpectedness", *Proceedings of Pacific Asia Conference on Knowledge Discovery in Databases (PAKDD)*, pp. 72-86. Melbourne, 1998.

[44] Duda, R.O., Hart, P.E. (1973). *Pattern Classification and SenseAnalysis* . John Wiley & Sons, Inc., New York NY.

[45] Duda, R.O., Hart, P.E., and Stork, D.G. (2001). *Pattern Classification*. 2nd Edition, John Wiley & Sons, Inc., New York NY.

[46] Dudoit S. and Fridlyand J., (2003) "Bagging to improve the accuracy of a clustering procedure", *Bioinformatics*, 19 (9), pp. 1090-1099, 2003

[47] Dumitrescu, D., Lazzerini, B., and Jain, L.C. (2000). *Fuzzy Sets and Their Application to Clustering and Training*. CRC Press LLC, New York.

[48] Efron, B. (1979) "Bootstrap methods: Another Look at the Jackknife". *Annals of Statistics*, 1979, 7: 1-26.

[49] El-Sheikh, E. M. (2001). *An Architecture for the generation of intelligent tutoring systems from reusable components and knowledge-based systems*, (Ph.D. Dissertation), 2001, MSU.

[50] Ester, M., Kriegel, H.-P., Xu. X. (1995) "A Database Interface for Clustering in Large Spatial Databases", *Proceedings of the Knowledge Discovery and Data Mining Conference*, pages 94-99, Montreal, Canada, 1995.

[51] eCollege, See http://convene.com; http://eCollege.com (real education); http://www.e-education.com (jones knowledge)

[52] Falkenauer E. (1997). *Genetic Algorithms and Grouping Problems.* John Wiley & Sons, 1998.

[53] Fayyad, U. M., Pitatesky-Shapiro, G., Smyth, P., and Uthurasamy, R. (1996). *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press.

[54] Fern, X., and Brodley, C. E., (2003) "Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach", In Proc. *20th Int. conf. on Machine Learning*, *ICML* 2003.

[55] Fischer, B. and Buhmann, J.M. (2002) "Data Resampling for Path Based Clustering", in: L. Van Gool (Editor), *Pattern Recognition - Symposium of the DAGM* 2002, pp. 206 - 214, LNCS 2449, Springer, 2002.

[56] Fischer, B. and Buhmann, J.M. (2003) "Path-Based Clustering for Grouping of Smooth Curves and Texture Segmentation", *IEEE Trans. on PAMI*, 25 (4), pp. 513-518, 2003.

[57] Frawley, W., Piatetsky-Shapiro, G., and Matheus, C. (1992). *Knowledge Discovery in Databases: An Overview*. AI Magazine, Fall 1992, pgs 213-228.

[58] Fred, A.L.N. (2001) "Finding Consistent Clusters in Data Partitions". In Proc. *3d Int. Workshop on Multiple Classifier Systems*. Eds. F. Roli, J. Kittler, LNCS, 2001, 2364: 309-318

[59] Fred, A.L.N. and Jain, A.K. (2002) "Data Clustering Using Evidence Accumulation", In Proc. of the *16th International Conference on Pattern Recognition*, *ICPR* 2002 ,Quebec City: 276 – 280.

[60] Freitas, A.A. (2002) "A survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery", In: A. Ghosh and S. Tsutsui. (Eds.) *Advances in Evolutionary Computation*, pp. 819-845. Springer-Verlag, 2002.

[61] Freitas, A.A. (1999) "On rule interestingness measures.", Knowledge-Based Systems journal 12 (5-6), 309-315. Oct. 1999.

[62] Freitas, A.A. (1999) *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Berlin: Springer-Verlag, 2002.

[63] Frossyniotis, D., Likas, A., and Stafylopatis, A., "A clustering method based on boosting", *Pattern Recognition Letters*, Volume 25, Issue 6, 19 April 2004, Pages 641-654

[64] Fridlyand, J. and Dudoit, S. (2001) "Applications of resampling methods to estimate the number of clusters and to improve the accuracy of a clustering methods," Division of Biostattistics, University of California, Berkley, Tech. Rep. 600, 2001.

[65] Fu Y. and Han, J., (1995) "Meta-rule-guided mining of association rules in relational databases". *Proc. 1995 Int'l Workshop. on Knowledge Discovery and Deductive and Object-Oriented Databases (KDOOD'95)*, Dec. 1995, pp. 39-46.

[66] Garofalakis, M., Rastogi, R. and Shim K. (1999). "SPIRIT: Sequential Pattern Mining with Regular Expression Constraints". In *Proc. of the 25th Very Large Database Conference*, Edinburgh, Scotland, September 1999.

[67] Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning,* MA, Addison-Wesley.

[68] *Goossens, M., Rahtz, S. and Mittelbach, F. (1997).* The LaTeX Graphics Companion, *Addison Wesley.*

[69] Guerra-Salcedo C. and Whitley D. (1999). "Feature Selection mechanisms for ensemble creation: a genetic search perspective". In: Freitas AA (Ed.) *Data Mining with Evolutionary Algorithms: Research Directions – Papers from the AAAI Workshop*, 13-17. Technical Report WS-99-06. AAAI Press, 1999.

[70] Guha, S., Rastogi, R., and Shim, K. (1998) "CURE: An efficient clustering algorithm for large databases" *In Proceeding of the 1998 ACM SIGMOD International Conference on Management of Data Engineering,* Pages 512-521 Seattle, WA USA.

[71] *Hall, M. A., and Smith, L.A. (1997), "Feature subset selection: a correlation based filter approach",* Proceeding of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems*, Springer, Page 855-858.*

[72] Han, J.; Kamber, M.; and Tung, A. 2001. Spatial Clustering Methods in Data Mining: A Survey. In *Miller, H., and Han, J., eds., Geographic Data Mining and Knowledge Discovery.* Taylor and Francis. 21

[73] Hand*, D. J. (1987),* Discrimination and Classification*, John Wiley & Sons, Chichester U.K.*

[74] Hastie, T., Tibshirani, R., Friedman, J.H. (2001) *The Elements of Statistical Learning*, Springer-Verlag, 2001

[75] Heckerman, D. (1996), *The process of Knowledge Discovery in Database*, 273-305, AAAI/MIT Press.

[76] *Harrell, Jr., Frank E. (2001),* Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis*, Springer-Verlag, New York.*

[77] Hoppner, F.; Klawnn, F.; Kruse, R.; Runkler, T.; (2000). *Fuzzy Cluster Analysis: "Methods for classification, data analysis and image recognition"*, John Wiley & Sons, Inc., New York NY.

[78] Houtsma M., and Swami A., (1993) "Set-oriented mining of association rules". Research Report RJ 9567, IBM Almaden Research Centers, San Jose California, 1993

[79] Hume, D. *An Enquiry Concerning Human Understanding,* Oxford Philosophical Texts, Tom L. Beauchamp (Editor) Oxford University Press, 1999

[80] Jain, A.K.; and Dubes; R. C. (1988). *Algorithms for Clustering Data.* Englewood Cliffs, NJ: Prentice-Hall.

[81] Jain, A.K.; Mao, J.; and Mohiuddin, K.; (1996). "Artificial Neural Networks: A Tutorial," *IEEE Computer*, March. 1996.

[82] Jain A.K. and Moreau (1987), "The Bootstrap Approach to Clustering", in *Pattern Recognition Theory and Applications*, P.A. Devijver and J. Kittler (eds.), Springer-Verlag, 1987, pp. 63-71.

[83] Jain, A.K.; Duin R. P.W.; and Mao, J. (2000). "Statistical Pattern Recognition: A Review", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 1, January 2000.

[84] Jain, A.K.; Murty, M.N.; Flynn, P.J. (1999) "Data clustering: a review", *ACM Computing Surveys*, Vol. 31, N.3, pp 264-323, 1999

[85] Jain, A.K.; Zongker, D.; (1997). "Feature Selection: Evaluation, Application, and Small Sample Performance" *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 2, February 1997.

[86] John, G. H.; Kohavi R.; and Pfleger, K.; (1994). "Irrelevant Features and Subset Selection Problem", *Proceeding of the Eleventh International Conference of Machine Learning*, page 121-129, Morgan Kaufmann Publishers, San Francisco, CA.

[87] Karhunen J., Oja E., Wang L., Vigario R., and Joutsenalo P. (1997) "A class of neural networks for independent component analysis", *IEEE Transactions on NeuRAL Networks.* Vol. 8, No. 3, pp486—504, 1997.

[88] Kashy, D A, Albertelli, G, Kashy, E, and Thoennessen, M, (2001), "Teaching with ALN Technology: Benefits and Costs", *Journal of Engineering Education,* Vol. 90, No. 4, October 2001, pp. 499-505

[89] Kashy, E, Gaff, S, J, Pawley, N, H, Stretch, W, L., Wolfe, S, L., Morrissey, D.J., Tsai, Y., (1995) "*Conceptual Questions in Computer-Assisted Assignments*", American Journal of Physics, Vol, No 63, 1995, pp. 1000-1005.

[90] Kashy, E., Thoennessen, M., Tsai, Y., Davis, N. E., and Wolfe, S. L. (1998), "Using Networked Tools to Promote Student Success in Large Classes", Journal of Engineering Education, ASEE, Vol. 87, No. 4, 1998, pp. 385-390

[91] Kashy, E., Thoennessen, M., Tsai, Y., Davis, N.E., and Wolfe, S.L. (1997), *"Using Networked Tools to Enhance Student Success Rates in Large Classes,"* Frontiers in Education Conference, Teaching and Learning in an Era of Change, IEEE CH 36099.

[92] Kaufman, L., and Rousseuw P. (1990) Finding Groups in Data- An Introduction to Cluster Analysis, Wiley Series in Probability and Mathematical Sciences, 1990.

[93] Klosgen, W., and Zytkow, J., (2002) *Handbook of Data Mining and Knowledge Discovery*, Oxford University Press, 2002.

[94] Kluger, A, N, DeNisi, A, (1996). "The Effects of Feedback Interventions on Performance: Historical Review, a Meta-Analysis and a Preliminary Feedback Intervention Theory". Psychological Bulletin, 119, 254-284.

[95] Kluger, A, N, DeNisi, (1998). A, "*Feedback interventions: Toward the understanding of a double-edged sword*", Current Directions in Psychological Science, 7, 67, 1998

[96] Kohavi, Ron (1995) "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", *Proceeding of International Joint Conference on Artificial intelligence (IJCIA)*, 1995.

[97] Kontkanen, P., Myllymaki, P., and Tirri, H. (1996), Predictive data mining with finite mixtures. In *Proceeding 2nd International Conference* "*Knowledge Discovery and Data Mining (KDD'96)*", pages 249-271, Portland, Oregon, August 1996.

[98] Kortemeyer, G., MSU LectureOnline software package, 1997-2000.

[99] Kortemeyer, G., and Bauer, W. (1999), "Multimedia Collaborative Content Creation (mc^3) - The MSU LectureOnline System", *Journal of Engineering Education*, 88 (4), 421.

[100] Kortemeyer, G., Albertelli, G., Bauer, W., Berryman, F., Bowers, J., Hall, M., Kashy, E., Kashy, D., Keefe, H., Minaei-Bidgoli, B., Punch, W.F., Sakharuk, A., Speier, C. (2003): "The LearningOnline Network with Computer-Assisted Personalized Approach (LON-CAPA)". *PGLDB 2003 Proceedings of the I PGL Database Research Conference,* Rio de Janeiro, Brazil, April 2003.

[101] Kotas, P, (2000) "Homework Behavior in an Introductory Physics Course", Masters Thesis (Physics), Central Michigan University (2000)

[102] Kuncheva , L.I., and Jain, L.C., (2000), "Designing Classifier Fusion Systems by Genetic Algorithms", *IEEE Transaction on Evolutionary Computation*, Vol. 33 (2000), pp 351-373.

[103] Mason, B, J, Bruning, R, "*Providing Feedback in Computer-based instruction: What the Research Tells Us*." http://dwb.unl.edu/Edit/MB/MasonBruning.html (2003).

[104] McLachlan, G.J. and Krishnan, T. (1997) *The EM Algorithm and Extensions*. Wiley series in probability and statistics.

[105] R. Meo, "Replacing Support in Association Rule Mining", Rapporto Tecnico RT70-2003, Dipartimento di Informatica, Università di Torino, April, 2003

[106] Murty, M. N. and Krishna, G. (1980). "A computationally efficient technique for data clustering". *Pattern Recognition. 12*, pp 153–158.

[107] Lange, R. (1996), An empirical test of weighted effect approach to generalized prediction using neural nets. In *Proceeding 2nd International Conference* "*Knowledge Discovery and Data Mining (KDD'96)*", pages 249-271, Portland, Oregon, August 1996.

[108] Lecture*Online,* See *http://www.lite.msu.edu/kortemeyer/lecture.html*

[109] Levine, E. and Domany E., (2001) "Resampling method for unsupervised estimation of cluster validity". *Neural Computation*, 2001, 13, 2573--2593.

[110] Loh, W.-Y. & Shih, Y.-S. (1997). Split Selection Methods for Classification Trees, Statistica Sinica 7: 815-840.

[111] Liu, B., Hsu, W., Mun, L.F. and Lee, H., (1999) "Finding Interesting Patterns Using User Expectations", *IEEE Transactions on Knowledge and Data Engineering*, Vol 11(6), pp. 817-832, 1999.

[112] Liu, B., Hsu, W. and Ma, Y, (2001) "Identifying Non-Actionable Association Rules", *Proc. Of seventh ACM SIGKDD International conference on Knowledge Discovery and Data Mining(KDD-2001)* San Francisco, USA.

[113] Lim, T.-S., Loh, W.-Y. & Shih, Y.-S. (2000). "A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms". *Machine Learning*, Vol. 40, pp. 203--228, 2000. See http://www.stat.wisc.edu/~limt/mach1317.pdf)

[114] LON-CAPA, See *http://www.lon-capa.org*

[115] Lu, S. Y., and Fu, K. S. (1978). " A sentence-to-sentence clustering procedure for pattern analysis." *IEEE Transactions on Systems, Man and Cybernetics SMC 8*, 381-389.

[116] Lu, H., Setiono, R., and Liu, H. (1995). Neurorule: A connectionist approach to data mining. *Proceeding 21st International Conference: Very Large Databases*, pages 478-489, Zurich, Switzerland, September 1995.

[117] Ma, Y., Liu, B., Kian C., Wong, Yu, P.S., and Lee, S.M., (2000) "Targeting the Right Students Using Data Mining". *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2000, Industry Track)*, Aug, 2000, Boston, USA

[118] Martin-Bautista MJ and Vila MA. (1999) A survey of genetic feature selection in mining issues. *Proceeding Congress on Evolutionary Computation (CEC-99)*, 1314-1321. Washington D.C., July 1999.

[119] Masand, B. and Piatetsky-Shapiro, G. (1996), A Comparison of approaches for maximizing business payoff of prediction models. In *Proceeding 2nd International Conference* "*Knowledge Discovery and Data Mining (KDD'96)*", pages 195-201, Portland, Oregon, August 1996.

[120] Matheus, C.J.. and Piatetsky-Shapiro, G. (1994), An application of KEFIR to the analysis of healthcare information. In *Proceeding* "*AAAI'94 Workshop Knowledge Discovery in Database (KDD'94)*", pages 441-452, Seattle, WA, July 1994.

[121] Michalewicz Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs.* 3rd Ed. Springer-Verlag, 1996.

[122] Michalski, R.S., Bratko, I., Kubat M. (1998), *Machine Learning and Data Mining, Methods and applications*, John Wiley & Sons, New York.

[123] Michalski, R.S., Stepp, R.E., III (1983), "Automated construction of classification: conceptual clustering versus numerical taxonomy" *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI 5,* 396-410.

[124] Mitchell, Tom M. (1997), *Machine Learning*, McGraw-Hill.

[125] Minaei-Bidgoli, B., Tan, P-N., Punch, W.F., (2004g) "Mining Interesting Contrast Rules for a Web-based Educational System", *International Conference on Machine Learning Applications (ICMLA 2004)*, Louisville, KY, USA, Dec 2004.

[126] Minaei-Bidgoli, B., Kortemeyer G., Punch, W.F., (2004f) "Association Analysis for an Online Education System", *IEEE International Conference on Information Reuse and Integration (IRI-2004)*, Las Vegas, Nevada, USA, Nov 2004.

[127] Hall, M., Parker, J., Minaei-Bidgoli, B., Albertelli, G., Kortemeyer, G., and Kashy, E., "Gathering and Timely Use of Feedback from Individualized On-line Work" *(IEEE/ASEE) FIE 2004 Frontier In Education*, Savannah, Georgia, USA, Oct. 2004.

[128] Minaei-Bidgoli, B., Kortemeyer, G., Punch, W.F., (2004e) "Optimizing Classification Ensembles via a Genetic Algorithm for a Web-based Educational System", *IAPR International workshop on Syntactical and Structural Pattern Recognition (SSPR 2004) and Statistical Pattern Recognition (SPR 2004)*, Lisbon, Portugal, Aug. 2004.

[129] Minaei-Bidgoli, B., Kortemeyer, G., Punch, W.F., (2004d) "Enhancing Online Learning Performance: An Application of Data Mining Methods", *The 7th IASTED International Conference on Computers and Advanced Technology in Education (CATE 2004)* Kauai, Hawaii, USA, August 2004.

[130] Minaei-Bidgoli, B., Kortemeyer, G., Punch, W.F., (2004c) "Mining Feature Importance: Applying Evolutionary Algorithms within a Web-Based Educational System", *International Conference on Cybernetics and Information Technologies, Systems and Applications: CITSA 2004*, Orlando, Florida, USA, July 2004.

[131] Minaei-Bidgoli, B., Punch, W.F., (2003) "Using Genetic Algorithms for Data Mining Optimization in an Educational Web-based System", *GECCO 2003 Genetic and Evolutionary Computation Conference*, Springer-Verlag 2252-2263, July 2003 Chicago, IL.

[132] Minaei-Bidgoli, B., Kashy, D.A., Kortemeyer G., Punch, W.F., (2003) "Predicting Student Performance: An Application of Data Mining Methods with an educational Web-based System", *(IEEE/ASEE) FIE 2003 Frontier In Education*, Nov. 2003 Boulder, Colorado.

[133] Minaei-Bidgoli, B., Topchy A., and Punch, W.F., (2004a) "Ensembles of Partitions via Data Resampling", to be appear in proceeding of IEEE International Conference on Information Technology: Coding and Computing, ITCC 2004, vol. 2, pp. 188-192, April 2004, Las Vegas, Nevada.

[134] Minaei-Bidgoli, B., Topchy A., and Punch, W.F., (2004b) “A Comparison of Resampling Methods for Clustering Ensembles”, in Proc. Intl. Conf. Machine Learning Methods Technology and Application, MLMTA 04, Las Vegas, 2004.

[135] McLachlan, G. (1992), *Discrimination Analysis and Statistical Pattern Recognition*, John Wiley & Sons, New York.

[136] McQueen, J. B. (1967). “Some methods of classification and analysis of multivariate observations.” *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281-297.

[137] Mori, Y.; Kudo, M.; Toyama, J.; and Shimbo, M. (1998), “Visualization of the Structure of Classes Using a Graph”, *Proceeding of International Conference on Pattern Recognition,* Vol. 2, Brisbane, August 1998, page 1724-1727.

[138] Montgomery, Douglas C., Peck, Elizabeth A., and Vining, Geoffrey G. (2001) *Introduction to linear regression analysis.* John Wiley & Sons, Inc., New York NY.

[139] Monti, S., Tamayo, P., Mesirov, J., Golub, T., (2003) “Consensus Clustering: A reamlping-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data”, *Journal on Machine Learning*, Volume 52 Issue 1-2, July 2003.

[140] Moore, Geoffrey, A, “*Crossing the Chasm*”, HarperCollins, 2002. Users of LON-CAPA in its very early beta versions would appreciate the statement on p.31, as these early adopters also forgave “…ghastly documentation…” as well as “… bizarrely obtuse methods of invoking needed functions …”

[141] Muhlenbein and Schlierkamp-Voosen D., (1993). Predictive Models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization, *Evolutionary Computation*, Vol. 1, No. 1, pp. 25-49, 1993

[142] Murray, T. (1996). “From Story Boards to Knowledge Bases: The First Paradigm Shift in Making CAI, Intelligent”, *ED-MEDIA 96 Educational Multimedia and Hypermedia Conference*, Charlottesville, VA.

[143] Murray, T. (1996). “Having It All, Maybe: Design Tradeoffs in ITS Authoring Tools”. *ITS '96 Third Intl. Conference on Intelligent Tutoring Systems*, Montreal, Canada, Springer-Verlag.

[144] Murthy, S. K. (1998), "Automatic construction of decision trees from data: A multidisciplinary survey”, *Data Mining and Knowledge Discovery,* vol. 4, pp. 345--389, 1998.

[145] MMP, Multi-Media Physics, See *http://lecture.lite.msu.edu/~mmp/*

[146] Ng, R.T., Han, J. (1994) "Efficient and Effective Clustering Methods for Spatial Data Mining", In *20th International Conference on Very Large Data Bases*, Pages 144-145, September 1994, Santiago, Chile.

[147] Odewahn, S.C., Stockwell, E.B., Pennington, R.L., Humphreys, R.M. and Zumach W.A. (1992). Automated Star/Galaxy Discrimination with Neural Networks, *Astronomical Journal*, 103: 308-331.

[148] PhysNet, See *http://physnet2.pa.msu.edu/*

[149] Park, B.H. and Kargupta, H. (2003) "Distributed Data Mining". In The *Handbook of Data Mining*. Ed. Nong Ye, Lawrence Erlbaum Associates, 2003

[150] Park Y and Song M. (1998). A genetic algorithm for clustering problems. *Genetic Programming 1998: Proceeding of 3rd Annual Conference*, 568-575. Morgan Kaufmann, 1998.

[151] Pascarella, A, M, (2004) "The Influence of Web-Based Homework on Quantitative Problem-Solving in a University Physics Class", NARST Annual Meeting Proceedings (2004)

[152] Pei, M., Goodman, E.D., and Punch, W.F. (1997) "Pattern Discovery from Data Using Genetic Algorithms", *Proceeding of 1st Pacific-Asia Conference Knowledge Discovery & Data Mining (PAKDD-97).* Feb. 1997.

[153] Pei, M., Punch, W.F., Ding, Y., and Goodman, E.D. (1995) "Genetic Algorithms For Classification and Feature Extraction", presented at the *Classification Society Conference* , June 95.

[154] See http://garage.cse.msu.edu/papers/papers-index.html

[155] Pei, M., Punch, W.F., and Goodman, E.D. (1998) "Feature Extraction Using Genetic Algorithms", *Proceeding of International Symposium on Intelligent Data Engineering and Learning'98 (IDEAL'98)*, Hong Kong, Oct. 98.

[156] Piatetsky-Shapiro G. and Matheus. C. J., (1994) "The interestingness of deviations". In *Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases*, pp. 25-36, 1994.

[157] Punch, W.F., Pei, M., Chia-Shun, L., Goodman, E.D., Hovland, P., and Enbody R. (1993) "Further research on Feature Selection and Classification Using Genetic Algorithms", In *5th International Conference on Genetic Algorithm* , Champaign IL, pp 557-564, 1993.

[158] Quinlan, J. R. (1986), Induction of decision trees. *Machine Learning*, 1:81-106.

[159] Quinlan, J. R. (1987), Rule induction with statistical data, a comparison with multiple regression. *Journal of Operational Research Society*, 38, 347-352.

[160] Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann.

[161] Quinlan, J. R. (1994), *Comparing connectionist and symbolic learning methods*, MIT Press.

[162] Roth, V., Lange, T., Braun, M., Buhmann, J.M. (2002) "A Resampling Approach to Cluster Validation", in:, Proceedings in Computational Statistics: 15th Symposium COMPSTAT 2002, Physica-Verlag, Heidelberg, 123-128.

[163] Ruck, D.W. , Rogers, S.K., Kabirsky, M., Oxley, M.E., and Suter, B.W. (1990), "The Multi-Layer Perceptron as an Approximation to a Bayes Optimal Discriminant Function"; *IEEE Transactions on Neural Networks*, vol. 1, no. 4.

[164] Russell, S.; and Norvig P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall.

[165] Shih, Y.-S. (1999), "Families of splitting criteria for classification trees", *Statistics and Computing*, Vol. 9, 309-315.

[166] Shute, V. and J. Psotka (1996). Intelligent Tutoring Systems: Past, Present, and Future. *Handbook of Research for Educational Communications and Technology*. D. Jonassen. New York, NY, Macmillan.

[167] Shute, V. J. (1991). "Who is Likely to Acquire Programming Skills?" *Journal of Educational Computing Research* 1: 1-24.

[168] Shute, V. J., L. A. Gawlick-Grendell, et al. (1993). *An Experiential System for Learning Probability: Stat Lady*. Annual Meeting of the American Educational Research Association, Atlanta, GA.

[169] Shannon, C.E., (1948) "A mathematical theory of communication", *Bell System Technical Journal,* vol. 27, pp. 379-423 and 623-656, July and October, 1948.

[170] Shute, V. J. and J. W. Regian (1990). *Rose garden promises of intelligent tutoring systems: Blossom or thorn?* Proceedings from the Space Operations, Applications and Research Symposium, Albuquerque, NM.

[171] Shute, V. R. and R. Glaser (1990). "A Large-scale Evaluation of an Intelligent Discovery World: Smithtown." *Interactive Learning Environments* 1: 51-76.

[172] Skalak D. B. (1994). Using a Genetic Algorithm to Learn Prototypes for Case Retrieval an Classification. *Proceeding of the AAAI-93 Case-Based Reasoning*

Workshop, pp. 64-69. Washigton, D.C., American Association for Artificial Intelligence, Menlo Park, CA, 1994.

[173] Siedlecki, W., Sklansky J., (1989), "A note on genetic algorithms for large-scale feature selection", *Pattern Recognition Letters,* Vol. 10, Page 335-347, 1989.

[174] Silberschatz A. and Tuzhilin, A., (1995) "On subjective measures of interestingness in knowledge discovery". *Proceeding of KDD*, 275-281, 1995.

[175] Silberschatz A. and Tuzhilin, A. (1996) "What makes patterns interesting in Knowledge discovery systems". *IEEE Transactions on Knowledge and Data Engineering*, 8(6):970-974, December, 1996.

[176] Singley, K. and J. R. Anderson (1989). The Transfer of Cognitive Skill. Cambridge, MA, Harvard University Press.

[177] Sleeman, D. (1987). PIXIE: A Shell for Developing Intelligent Tutoring Systems. *Artificial Intelligence and Education,* R. W. Lawler and M. Yazdani. Norwood, NJ, Alex Publishers**:** 239-265.

[178] Sleeman, D. H. and J. S. Brown (1982). *Intelligent Tutoring Systems,* London, UK, Academic Press.

[179] Srikant, R. and Agrawal R. (1996). "Mining Quantitative Association Rules in Large Relational Tables", In *Proc. of the ACM-SIGMOD 1996 Conference on Management of Data*, Montreal, Canada, June 1996

[180] Strehl, A. and Ghosh, J. (2002) "Cluster ensembles - a knowledge reuse framework for combining multiple partitions". *Journal on Machine Learning Research*, 2002, 3: 583-617

[181] Tan, P.N., Kumar V., and Srivastava, J. (2004), "Selecting the Right Objective Measure for Association Analysis", *Information Systems*, 29(4), 293-313, 2004.

[182] Tobias, S., Raphael, J., (1997) "The Hidden Curriculum", contribution by E. Kashy and D. J. Morrissey, pp 165-167, Plenum Press, New York 1997.

[183] Topchy, A., Minaei-Bidgoli, B., Jain, A.K., Punch, W.F., (2004) "Adaptive Clustering Ensembles", *International Conference on Pattern Recognition (ICPR 2004)*, Cambridge, UK, Aug. 2004.

[184] Topchy, A., Jain, A.K., and Punch W.F. (2003a), "Combining Multiple Weak Clusterings", In proceeding of *IEEE Intl. Conf. on Data Mining* 2003.

[185] Topchy, A., Jain, A.K., and Punch W.F. (2003b), "A Mixture Model of Clustering Ensembles", submitted to *SIAM Intl. Conf. on Data Mining* 2003

[186] TopClass ^{TM}, WBT Sytems, San Fransisco, CA. (TopClass is a trademark of WBT Systems, see *http://www.wbtsystems.com*).

[187] Toussaint, G. T. (1980). “The relative neighborhood graph of a finite planar set.” *Pattern Recognition 12*, 261-268.

[188] Trunk, G.V. (1979), “A problem of Dimensionality: A Simple Example”, *IEEE Transaction on Pattern Analysis and Machine Intelligence,* Vol. PAMI-1, No. 3, July 1979.

[189] Urban-Lurain, M. (1996). Intelligent Tutoring Systems: *An Historic Review in the Context of the Development of Artificial Intelligence and Educational Psychology*, Michigan State University.

[190] Urquhart, R. (1982). “Graph theoretical clustering based on limited neighborhood sets.” *Pattern Recognition 15*, 173-187.

[191] Vafaie H and De Jong K. (1993). Robust feature Selection algorithms. *Proceeding 1993 IEEE Int. Conf on Tools with AI*, 356-363. Boston, Mass., USA. Nov. 1993.

[192] VU, (Virtual University), See *http://vu.msu.edu/*; *http://www.vu.org*.

[193] Wu, Q., Suetens, and Oosterlinck, A. (1991) Integration of heuristic and Bayesian approaches in a pattern-classification system. In Piatetsky-Shapiro G., and Frawely W.J., editors, “*Knowledge Discover yin Database*”, pages 249-260, AAAI/MIT press.

[194] WebCT ^{TM}, University of British Columbia, Vancouver, BC, Canada. (WebCT is a trademark of the University of British Columbia , see *http://www.webct.com/*)

[195] Weiss, S. M. and Kulikowski C. A. (1991), *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*, Morgan Kaufman.

[196] Woods, K.; Kegelmeyer Jr., W.F.; Bowyer, K. (1997) “Combination of Multiple Classifiers Using Local Area Estimates”; *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 4, 1997.

[197] Yazdani, M. (1987). Intelligent Tutoring Systems: An Overview. *Artificial Intelligence and Education: Learning Environments and Tutoring Systems*. R. W. Lawler and M. Yazdani. Norwood, NJ, Ablex Publishing. **1:** 183-201

[198] Zaïane, Osmar R. (2001) “Web Usage Mining for a Better Web-Based Learning Environment”, in Proc. of *Conference on Advanced Technology for Education*, pp 60-64, Banff, Alberta, June 27-28, 2001.

[199] Zhang, B., Hsu, M., Forman, G. (2000) "Accurate Recasting of Parameter Estimation Algorithms using Sufficient Statistics for Efficient Parallel Speed-up Demonstrated for Center-Based Data Clustering Algorithms", *Proc. 4th European Conference on Principles and Practice of Knowledge Discovery in Databases, in Principles of Data Mining and Knowledge Discovery*, D. A. Zighed, J. Komorowski and J. Zytkow (Eds.), 2000.

[200] Zhang Fern, X., and Brodley, C. E. (2003) "Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach", in Proc. of the *20th Int. conf. on Machine Learning ICML* 2003.

[201] Zhang, T; Ramakrishnan, R., and Livny, M. (1996), "BIRCH: An Efficient Data Clustering Method for Very Large Databases", Proceeding of the *ACM SIGMOD Record*, 1996, 25 (2): 103-114