

83826

**SUPERVISED and UNSUPERVISED LEARNING
TECHNIQUES in DATA MINING**

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of
Dokuz Eylül University
in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Computer Engineering

by
Mehmet Seval KAYGULU

DEÜ YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

July, 1999
İZMİR

83826

M.Sc. THESIS EXAMINATION RESULT FORM

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as thesis for the degree of Master of Science.



Assoc. Prof. Dr. Alp KUT
(Advisor)



Assist. Prof. Dr. Yalçın ÇEBİ

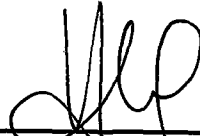
(Committee Member)



Assist. Prof. Dr. Adil ALPKOÇAK

(Committee Member)

Approved by the
Graduate School of Natural and Applied Sciences



Prof. Dr. Cahit HELVACI

Director

ACKNOWLEDGMENTS

My biggest thanks goes to Prof. Dr. Esen ÖZKARAHAN, Dr. Alp KUT, Dr. Yalçın ÇEBİ and Dr. Adil ALPKOÇAK. Without their knowledge and encouragement I could not be able to finish this work with this quality. They were not only academic advisors, but were like friends who support me in everything I do.

I will always be grateful to Prof. Dr. Esen ÖZKARAHAN since he taught me how to make research. I would like to thank to Dr. Adil ALPKOÇAK who help me for reference books. I would also like to thank to Dr. Yalçın ÇEBİ as a member of the thesis committee for his valuable comments.

Mehmet Seval KAYGULU

ABSTRACT

Large databases are composed of a lot of information some being very explicit and some being not so straight forward. At this point, data mining with all its techniques emerges to help us getting the most out of the databases. Since we can create a lot of relationships within the databases, it is of prime importance to choose the meaningful and useful ones, techniques like machine learning is used and their reliability is checked through statistical techniques.

In this study, knowledge discovery in databases, types of learning, and several data mining techniques are defined. First of all, the concept of knowledge discovery in databases and types of learning are briefly explained to enlighten the minds then, data mining is examined and discussed to a certain depth.

ÖZET

Büyük boyuttaki veri tabanları bir çok bilgiyi içerir. Bu bilgilerin bazıları açık ve kolay anlaşılır bilgilerdir. Ancak bazı bilgiler veri tabanı içinde gizlidir ve doğrudan ulaşılamazlar. İşte bu noktada, veri madenciliği (data mining) ve onun teknikleri kullanılarak, veri tabanından gizli bilgilerin büyük bir çoğunluğu elde edilebilir. Veri madenciliği yöntemleri ile pek çok ilişki elde edilebilir. Burada önemli olan, bu ilişkilerden anlamlı ve faydalı olanlarını seçebilmektir. Bu işlem için tümevarım veya suni zeka teknikleri kullanılabilir. Bu ilişkilerin güvenilirliği ise istatistiksel yöntemler kullanılarak test edilebilir.

Bu çalışmada veri tabanından bilgi keşfetme, öğrenme teknikleri ve veri madenciliğinin teknikleri üzerinde durulmuştur. Önce, veri tabanından bilgi keşfetme ve öğrenme teknikleri kısaca hatırlatılmıştır. Daha sonra ise, veri madenciliği konusu daha derinlemesine incelenmiş ve tartışılmıştır.

CONTENTS

	Page
Contents	VI
List of Tables	X
List of Figures	XI

Chapter One INTRODUCTION

1.1. Why do we need data mining?.....	1
1.2. Overview to knowledge discovery in databases process.....	2

Chapter Two TYPES OF LEARNING

2.1. Inductive learning.....	6
2.1.1. Models.....	6
2.1.1.1 Environment.....	7
2.1.1.2 Classes.....	8
2.1.1.3. Model transition function.....	8
2.1.1.4. Correctness of the model.....	9
2.2. Learning.....	10

2.2.1. Supervised learning.....	10
2.2.2. Unsupervised learning.....	11
2.3. Quality.....	11
2.4. Machine Learning.....	11

Chapter Three

DATA MINING

3.1. The comparison of machine learning with data mining.....	13
3.2. The training set.....	15
3.3. Classification.....	17
3.4. Clustering.....	18
3.4.1. Pattern matrix.....	20
3.4.2. Proximity matrix.....	21
3.4.3. Clustering methods and algorithms.....	23
3.4.3.1. Hierarchical clustering.....	23
3.4.3.2. Partitional clustering.....	27
3.5. Rules.....	30

Chapter Four

SEARCH ALGORITHMS

4.1. Search space.....	33
4.1.1. Description space.....	33
4.1.2. Operations.....	34
4.1.3. Domains of the attributes	35
4.1.4. Quality function.....	35
4.2. Search algorithm.....	39
4.2.1. Initial description.....	39

4.2.1. Search graph.....	40
4.3. Heuristic search.....	42
4.3.1. Hill climber.....	42
4.3.2. Limitations on the operations.....	42

Chapter Five

PROBLEMS

5.1. Limited information.....	44
5.1.1. Incomplete information.....	44
5.1.2. Sparse data.....	45
5.2. Data corruption.....	45
5.2.1. Noise.....	45
5.2.2. Missing attribute values.....	46
5.3. Databases.....	46
5.3.1. Size of database.....	47
5.3.2. Updates.....	48

Chapter Six

KNOWLEDGE REPRESENTATION

6.1. Propositional-like representations.....	49
6.1.1. Decision tree.....	50
6.1.2. Production rules.....	50
6.1.3. Decision list.....	51
6.1.4. Ripple-down rule sets.....	51
6.2. First order logic.....	52
6.3. Structured representations.....	53
6.3.1. Semantic nets.....	53
6.3.2. Frames and schemata.....	54

Chapter Seven
SAMPLE PROGRAM

7.1. Introduction.....	56
7.2. Database.....	56
7.3. Rules and classes.....	57
CONCLUSIONS.....	61
REFERENCES.....	62



LIST OF TABLES

	Page
Table 3.1 Properties of some animals.....	16
Table 6.1. An example of a frame.....	54



LIST OF FIGURES

	Page
Figure 2.1. Classification function P , state transition function τ and model transition function τ'	9
Figure 3.1. Diagram for machine learning.....	14
Figure 3.2. Diagram for data mining.....	14
Figure 3.3. Clusters of points pattern into two dimensions.....	20
Figure 4.1. A search graph.....	41
Figure 6.1. An example for the decision tree.....	50
Figure 6.2. An example of semantic nets.....	53
Figure 7.1. The decision tree formed by the program when attribute 'GELIR' is chosen.....	58
Figure 7.2. The decision tree formed by the program when attributes 'GELIR' and 'OKUL' are chosen.....	59
Figure 7.3. The decision tree formed by the program when attributes 'GELIR', 'OKUL', 'HUKBAS' and 'ANAYASA' are chosen.	60

CHAPTER ONE

INTRODUCTION

A database is a store of non-trivial information. Most important purpose of a database is the efficient retrieval of information. This retrieval information can be a copy of information stored in a database. Some important information can be hidden in a database, so that, retrieval information must be inferred this hidden information from the database. This information is not only statistical, but a relation between attributes of the database.

Data mining is the automatic process of handling of information from databases which can not be seen directly. Data mining is used for finding useful trends and patterns. In some articles and documents, the term data mining is given the same meaning with knowledge discovery in databases (KDD), means an automatic process of non-trivial extractions, formerly unknown and useful information (including rules, constrains and regularities) from data in databases. There are many other terms, with similar meaning or small difference in meaning, such as knowledge mining from databases, knowledge extraction, data archaeology, data dredging, data analysis, etc. Some researchers (U. Fayyad, G. Piatetsky-Shapiro, P.Smyth) suggest that data mining is only one of the steps of KDD.

1. Why do we need data mining?

Traditionally, analysts used manual process for analysis. Analysts, familiar with the data and using statistical techniques, have generated reports. But now, this

process is very difficult, expensive and slow because of rapid growth of data and increasing number of attributes in databases. On the other hand, this process, at the same time, is subjective.

We can store and access to reliable data efficiently and inexpensively with using current hardware and database technology. In raw form, datasets about business management, government administration, medicine, science or engineering have little value. Databases are calm resources that have potential to yield important benefits.

Who could organize billions of records, each having tens or hundreds of fields and, extract knowledge from such databases? These processes are over the human ability. We need new techniques and tools for knowledge discovery or extraction in databases.

2. Overview to knowledge discovery in databases process

In this section, it is accepted that knowledge discovery from databases (KDD) includes all steps for finding useful patterns in data. Data mining is only a particular step of KDD. Other steps are data preparation and selection, data cleaning, incorporation of prior knowledge, and interpretation of the result of mining. Random applications of data mining methods as data dredging can be wrong because of extraction of meaningless patterns.

KDD has evolved, and continues to evolve, from the intersection of research in such fields as databases, machine learning, pattern recognition, statistics, artificial intelligence and reasoning with uncertainty, knowledge acquisition for expert systems, data visualization, machine discovery, information retrieval, and high-performance computing. KDD software systems incorporate theories, algorithms, and methods from all of these fields. (Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., 1996, p.29)

KDD is interested in the all processes of knowledge discovery from datasets, such that how data is stored (types of database such as relational database, hierarchical database, network database, types of data such as numbers, text, images and voice) , which algorithms can be chosen that run efficiently on huge data sets. how results can be interpreted, how interpreted results can be visualized and how user interface can be modelled. And also, KDD interests in noise in data sets.

Statistics has much related with KDD. Handling patterns and knowledge inference has been a component of statistics. A statistician can find patterns if the statistician searches in any dataset sufficiently. These patterns can be seen significant from the view of statistics. But they are not significant in real world. To find nontrivial pattern is very important for KDD. Activity of understanding how to find these patterns correctly is data mining. KDD includes larger views of modeling than statistics. Aim of KDD is to provide tools that whole process of data analysis can be done . U. Fayyad defined the KDD process as: “The nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data ” (Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., 1996, p.29)

In above definition *data* is a set of facts, *pattern* is a description of a subset of the data. The steps of data preparation, search for patterns, knowledge retrieval and refinement (all these steps in multiple iteration) are named as *process*. The process is assumed to be *nontrivial*. The found pattern should be *valid* for new data. It is preferable that these patterns is *novel* at least to the system and to the user, and *potentially useful* for the user. And the patterns should be *understandable*.

There are many interactive and iterative (because, user can make many decisions) steps.

i. *Learning the application domain:*

In this step, analyst should search and understand prior knowledge and the aim of the application.

ii. *Creating a target dataset:*

In this step, analyst should select a dataset or data samples on which knowledge discovery is performed.

iii. Data cleaning and preprocessing:

In this step, analyst should remove noise, collect the necessary information to model, decide on strategies for getting missing data and decide database management system (DBMS) problems such as data types, schema and mapping of unknown values.

iv. Data reduction and projection:

In this step, analyst should find features to represent the data and, reduce the number of variables under consideration or to find constant representations for data, depending on the aim of the application.

v. Choosing the function of data mining:

In this step, analyst should decide the purpose of the model such as summarization, classification, regression, and clustering, that is derived by the data mining algorithm.

vi. Data mining:

In this step, analyst should decide which models and parameters are used for patterns in the data.

vii. Choosing the data mining algorithm:

In this step, analyst should search for patterns in the data in a particular representational form.

viii. Interpretation:

In this step, analyst should interpret the extracted patterns including visualization of these patterns and translating into terms understandable by users.

ix. Using discovered knowledge:

Incorporating the discovered knowledge into the performance system, trying to find out the conflicts between the knowledge acquired and the one previously extracted and taking actions related with the knowledge which take place in making use of the discovered knowledge.

CHAPTER TWO

TYPES OF LEARNING

The first purpose of database which is store of true information, is to retrieve efficient and useful knowledge. This knowledge sometimes can be of hidden form. Therefore, we must have some techniques to infer that hidden knowledge. From a logical point of view, there are two techniques to infer knowledge.

The first one of these two techniques is *deduction*. Inferred knowledge by deduction technique is a logical consequence of the information in the database. For example, each teacher gives a course and each student takes a course. We can infer the list of names of students and their teacher's names. This knowledge can be inferred from the database with applying the join operation between two relational tables such as TEACHER-COURSE and STUDENT-COURSE.

The second is *induction* technique. Generalized information can be inferred from the information in the database. For example, the knowledge " each student has at least one teacher " might be inferred from TEACHER-COURSE and STUDENT-COURSE relational tables. This is higher-level knowledge than inferred knowledge from the database by induction technique. If we can formulate this higher-level knowledge, we can predict the value of an attribute in terms of other attributes.

The knowledge inferred by induction technique may not be always true in the real world, it is only supported by the database. By the knowledge inferred by deduction technique is probably correct in the real world that is provided that the database is correct. Therefore we must carefully select the regularities that they are plausible and supported by the database.

1. Inductive learning

Humans try to understand their environment by simplifying it. Simplification of this environment is called a *model*. *Inductive learning* is the process of creation of a model of environment. During this process, cognitive system observes its environment. It recognizes similarities among objects and events in this environment. Cognitive system uses similar objects to make a class. It constructs rules for the behaviour of the members of a class. The set of rules of a class is called *class description*.

There are mainly two learning techniques. In *supervised learning*, classes are defined and examples of each class are given to the cognitive system by someone, let's say a teacher. The system will construct the class descriptions by discovering common properties in the examples for each class. The form 'if < description > then < class >' is called a *classification rule*. Classification rules can be used to predict the class of new, previously unseen objects. This inductive learning technique is also known as *learning from examples*.

In *unsupervised learning*, there is not any teacher. Cognitive system has to discover classes and their descriptions itself. System observes its environment and recognizes common properties of objects. This inductive learning technique is also known as *learning from observation and discovery*.

1.1. Models

Inductive learning is the process of creation of a model of the environment of the cognitive system. This model consists of classes which represent objects that have similar properties, and rules that describe properties of members of each class and changes in environment. Cognitive system uses the models to predict changes in the environment, and to interact with this environment.

1.1.1. Environment

Defining of environment depends on the context. Environment of a cognitive system may be defined in local terms such as students of a faculty, a football team, a chess board, or as the whole of the universe which includes the system itself.

The situation of the environment is described by a *state* S_t , at a specific time t . This state, S_t , has some rules which describe the properties of the objects in the environment and mutual relationships among the objects. But the state of the environment changes over time. At the time $t+1$, state S_{t+1} may have new objects and relationships, or some objects may have disappeared, and properties of objects may have changed. So that, we must have a function that describes how the environment changes over time. This function is called *state transition function* and it is represented by τ . Transition function maps from one state to another state.

Marcel Holsheimer and Arno Siebes have given a definition for the environment: "The environment is a state transition system, ie., a pair (S, τ) , where S is the set of all possible states and τ is the function $\tau: S \rightarrow S$. τ defines the next state S_{t+1} for any state S_t " (Holsheimer , M. & Siebes, A. p. 11)

EXAMPLE:

Assume that the state consists of a single object, with properties " name is Ali " and " second year student ". In the next state " name " of the object remains unchanged, but the property " second year student " has changed to " third year student " obeying the law that all second year students will be third year students if they achieve their courses at the end of the academic year.

To make a reliable internal copy of this state transition system is a straightforward way to create a model of the environment. Simply, all encountered states are stored and all transitions are recorded. The current state is compared with all stored states to predict the next state from the current state. But, this representation is suitable for simple environments that have a small number of various states. Otherwise, for realistic environments, the enormous amount of storage is needed to represent all

possible states that the current state will exactly match any of the previous states. And some times, it will be impossible to determine the all possible states.

Because of such difficulties, we must use abstractions instead of making a faithful internal copy of any state transition system. For abstraction, a small number of properties to characterize the objects in a state is used. Objects having the same subset of properties are mapped to the same internal representation.

1.1.2. Classes

We describe the state in the model with using a small number of properties. This may cause that distinct objects in the environment may be accepted as the same object. That means we collect the objects having same chosen properties in a group. This group is called *equivalence classes* of objects. The *class description* consists of the unique values of properties of the objects. To each class corresponds a class description.

We can construct a *classification function* $P:S \rightarrow C$ where C denotes the set of all classes, and to each class C_i corresponds a description D_i . The classification function P maps an object O in state S to class C_i if properties of O have the same values in the description D_i .

1.1.3. Model transition function

State transition function τ determines how the environment changes with respect to time in the real world. Cognitive system uses models instead of the real world. So that, the system should construct a model transition function τ' . Model transition function acts on class (or internal) representations similarly as state transition function acts on states. The model transition function τ' describes how the class (internal) representations $C_{t,i}$ of a state S_t of environment leads to the class representations $C_{t+1,i}$ of a state S_{t+1} .

Figure 2.1 shows all functions, we have talked about. An environmental state S_t changes to a state S_{t+1} . Class C_t is an internal representation of state S_t . Model transition function is applied to class C_t and C_t changes to a class C_{t+1} which is the internal representation of state S_{t+1} . As a result of application of the classification function P to state S_t , may occur more than one class, but one class C_t is chosen for simplification.

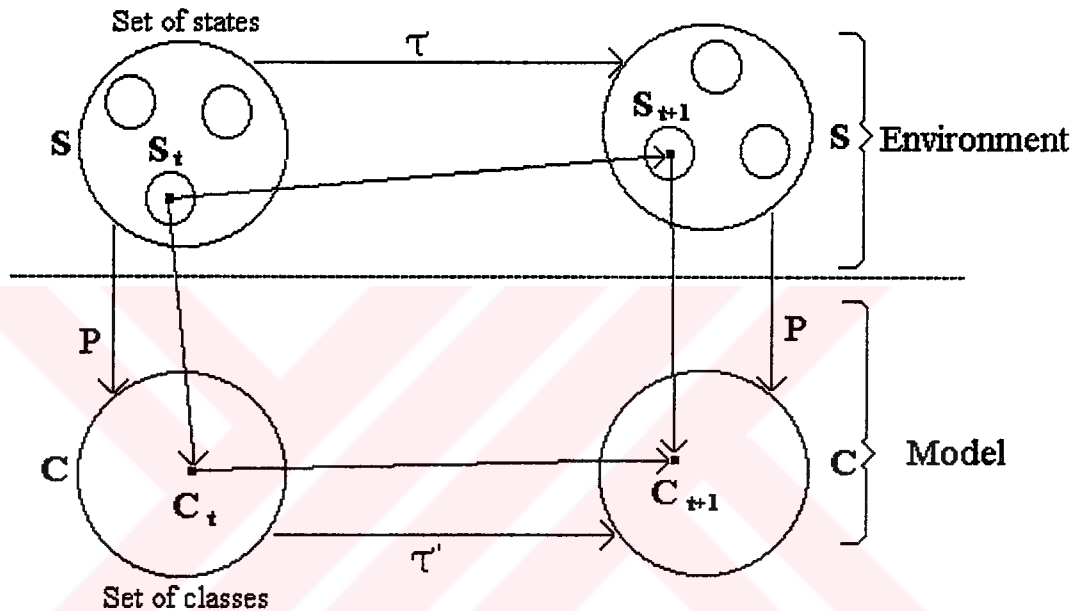


Figure 2.1: Classification function P , state transition function τ and model transition function τ'

1.1.4. Correctness of the model

If a model predicts the class representation of the next state correctly, this model is correct. This means that if the class representation of the next state at any time t , is identical to the predicted representation by the model. In other words;

$$P(\tau(S_t)) = \tau'(P(S_t))$$

2. Learning

The cognitive system should adapt itself to its environment. This means that the system should *learn*. Learning is to find suitable classes (internal representation) and a model transition function that acts on these classes. There are various learning strategies such as *learning by being told* and *learning from analogy*. For example, in learning by being told, a teacher acquires the knowledge like a textbook. The system only translates this knowledge to an internal format. In learning from analogy, the system changes existing rules to generate new rules which are applicable to new, similar situations. In inductive learning, there are mainly two strategies; supervised learning and unsupervised learning.

2.1. Supervised learning

In supervised learning or learning from examples, the teacher defines the classes and supplies pre-classified objects of each class. The system should only find the description for each class to construct the model. A *single class* or *multiple classes* can be defined by the teacher.

In single class learning, only one class C is defined by a teacher. This teacher also provides all examples. If an example is a member of the class, this example called *positive* example, otherwise it called *negative* example. Teacher may provide all positive examples. Or both positive and negative examples are provided. The negative examples can be seen as members of many other classes. With characteristic *description*, positive examples, members of class C are separated from negative examples which are not instances of class C .

In multiple classes learning, a finite number of classes C_1, C_2, \dots, C_n are defined by a teacher. Characteristic description D_i distinguishes positive examples of C_i from other examples (negative examples). Alternatively, the system constructs discriminating description that cover all objects. Discriminating description distinguishes an instance of a class from the instance of all other classes.

2.2. Unsupervised learning

In unsupervised learning or learning from observation and discovery, there is no teacher that defines the classes. The system has to find its own classes. In practice, the system has to construct some clusters of the set of states in the environment. Such as in supervised learning, objects or examples are known. The system has to observe the objects and constructs class descriptions or patterns. Class descriptions are constructed for each discovered class. Discovered classes cover all objects in the environment. Set of class descriptions is the result of unsupervised learning process.

3. Quality

Created model may change with respect to set of examples, and multiple models can be constructed from the same set of examples. It can be said that all created models can be correct with respect to given set of examples. Models should correctly predict the next state $S_{t+1,i}$ for all environmental states $S_{t,i}$ that already known. Models should be used also for any unseen state when new, unseen states occur.

Discovered or apparent relationships among states are not generally valid. Because the number of objects is limited. So that, apparent relationships can be different from really existing relationships among states in the environment.

The correctness of a model is not verified by checking for all possible states, because the number of possible states is infinite for most environments. If multiple models are constructed, the simplest model can be chosen. Because the simplest model is more likely to handle the nature of the phenomenon. (Ockham's razor rule)

4. Machine learning

Computers can be used for inductive learning processes . This process is called machine learning. Machine learning systems use a coded form of a finite set of

examples and observations, and do not interact directly with its environment. This coded finite set is called *training set*. In supervised learning, classes are defined by a user and system searches descriptions for each class. In unsupervised learning, machine learning system constructs the set of new discovered classes and class descriptions.



CHAPTER THREE

DATA MINING

The methods for handling regularities and rules are named data mining when the data set is a database. The knowledge (data) stored by a database has a different purpose than a learning process. This data may have noise and some values of attributes may be lost. To discover descriptions from a database is harder than machine learning where the ideal conditions has already being defined. Because of the size of database, the cost of inferring rules and verification of hypotheses is high. This cost can be reduced by using browsing optimization and caching. To remove noisy and missing values, statistical techniques are used.

As it has already being seen, we can say that learning is the process to construct the rules for transitions from state S_t to state S_{t+1} , based on objects in the environment and observations of states of the environment. Machine learning is an automatic learning process which uses computer. Machine learning systems use training set, instead of real environment. Automatic inductive learning process is called *data mining* when the training set is a database. Now we can say that data mining is a special kind of machine learning.

1. The comparison of machine learning with data mining

In machine learning, environment represents a finite set of objects. These objects are encoded in some readable form for the machine by the encoder. The set of encoded objects is the training set for machine learning algorithm, as shown in the figure 3.2.

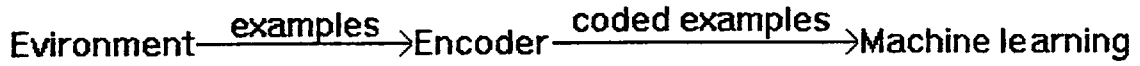


Figure 3.1: Diagram for machine learning

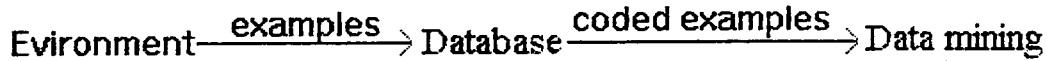


Figure 3.2: Diagram for data mining

In data mining, database is used instead of encoder (figure 3.2). Database consists of facts which are taken from the environment. It can be said that database is a small and simple model of the environment, because it has finite set of examples. Each state of the database represents a state of that environment. Each state transition of database represents a state transition of that environment. And data mining algorithm constructs a model from the database. That means, data mining algorithm infers classification rules that manage the classes of database objects. The rules for transition between classes should also be inferred from the transition in the database.

It may be seen that data mining and machine learning both have a similar framework. But there are important differences between them. First of all, in machine learning, training set is chosen suitable to help the learning process. On the other hand, database is not designed to help the data mining process. Objects of the database are chosen for the needs of applications. These objects may not meet the needs of data mining. In data mining, some attributes (or properties) may be chosen to simplify the learning process, but these attributes may not be in the database.

As a second and important difference, databases can contain some errors. In machine learning, learning algorithm often uses suitable examples which are chosen carefully and do not contain error or noise, but data mining algorithm has to cope with data which has noisy and contradictory data.

2. The training set

The training set of the learning algorithm of the data mining is database which contains non-trivial knowledge from the environment. There are many types of databases, that database management systems (DBMS) support. We are interested in relational type of database. In a relational database, examples (or objects or instances) are represented by tuples and properties of objects which are called attributes. Each tuple may have many attributes.

Each tuple can represent one or more objects in the environment. If tuples have at least one unique attribute, each tuple can represent only one object. Otherwise, each tuple may represent more than one object.

We can construct more than one table with using attributes and tuples. Each column of a table represents an attribute and each row of a table represents an example or object. We can recognize relationship between tables with common attribute (or attributes). Common attributes can be used for JOIN operations. In these tables, values of attributes may be NULL or unknown. If we use *Universal Relation Assumption*, we construct a single table which contains all objects and their properties. Of course, the values of attributes in this table may be NULL or unknown.

Each attribute or property of objects is an element of set A , $A = \{ A_1, A_2, \dots, A_n \}$. Distinct values of attributes form domains of attributes. We can say that domain of attribute A_1 is D_1 , domain of attribute A_2 is D_2 , and so on. Constructed table with attributes and their domains are called *training set*. All relations over attributes of the table is called *Universe U*. That means, $U = D_1 \times D_2 \times \dots \times D_n$. Now, we can say that, each training set is a finite subset of Universe U . Of course, we assume that each domain of attributes is finite and as a result, Universe is also finite.

Table 3.1: Properties of some animals

Name	Organ	Transportation	Ismamma
Stork	Wings	Fly	No
Hawk	Wings	Fly	No
Ostrich	Wings	Run	No
Penguin	Wings	Swim	No
Bat	Wings	Fly	Yes
Shark	Fins	Swim	No
Salmon	fins	Swim	No
Sea turtle	Legs	Swim	No

EXAMPLE :

We try to show properties which are explained in above paragraph with using table 3.1 shown above.

In table 3.1, objects are animals. Each row (tuple) explains properties of each animal. Each column (attributes) has a single property of an animal. First attribute is the name of animal. Second attribute is name of organ which that animal has. Third attribute is type of transportation which that animal uses. Fourth attribute describes whether the animal is a mammal or not. The attribute set is $A = \{ \text{Name, Organ, Transportation, Ismammal} \}$. Domain of first attribute ' Name ' is $D_1 = \{ \text{Stork, Hawk, Ostrich, Penguin, Bat, Shark, Salmon, Sea turtle} \}$, domain of second attribute ' Organ ' is $D_2 = \{ \text{Wings, Fins, Legs} \}$, domain of third attribute ' Transportation ' is $D_3 = \{ \text{Fly, Run, Swim} \}$ and finally, domain of fourth attribute ' Ismammal ' is $D_4 = \{ \text{Yes, No} \}$.

3. Classification

Data mining accepts the database as the environment. So that, the function of the data mining is to infer a model from databases. If a user prefers the supervised learning technique, that user defines a class or classes in the database. These classes are also called *concepts*. Some attributes (there may be only one attribute) in the database can be used to denote the class of a tuple. These attributes are called the *predicted attributes*. The other attributes which are different from predicted attributes are called *predicting attributes*.

A condition is a combination of values of the predicted attributes. A class is defined by using condition on the attributes. A class C_i is a subset of the training set S which is the database itself in data mining. If c_i is the condition for class C_i , all objects that satisfy the class condition c_i belong to class C_i . In other words;

$$C_i = \{ o \in S \mid \text{Values of } o \text{ satisfy condition } c_i \text{ (or } c_i(o) \text{) } \}$$

where o represents an object.

Objects are called *positive examples* or *instances* of the class C_i when objects satisfy condition c_i . Other objects of the training set S which do not satisfy condition c_i are called *negative examples*. Negative examples of C_i may belong to other class if there is more than one class.

There are different techniques to define classes by using predicted attributes. Any technique can be chosen depending on the data mining purpose or the databases used in data mining.

Conditions on attributes:

In this case, the combination of values of predicted attributes is used to denote classes. Sometimes, predicting attributes can be used with predicted attributes.

In table 3.1, we can define a class ' wings ' as a condition ' Organs = wings ', where organs is a predicted attribute. Any tuple or example which the attribute organs of this tuple has value ' wings ', belongs to class ' wings '. Or we can define

a class ' bird ' in table 3.1, as a condition ' Organs = wings ' and ' Ismammal = no ' where ' Organs ' is a predicted attribute but " Ismammal " is a predicting attribute.

Multiple databases:

Composed form of the multiple databases can be used as a training set. For example, we can search for differences between students for Law Faculty of Dokuz Eylül University and students of Law Faculty of Ankara University. All students which are stored in the database in Dokuz Eylül University, belong to the class " DEU " and, all students which are stored in the database in Ankara University, belong to the class " AU ".

Databases over time:

We can search global changes over a period. We use two training sets. These training sets are taken from the same database at different times. The training set at time t_i is S_i and at time t_j is S_j . All members of S_i belong to class " Time = i " and all members of S_j belong to class " Time = j ".

The training sets S_i and S_j can represent the same objects at different time. Some attributes of the same object may have different values. We can define a class in terms of S_i and S_j . We can use S_i as the training set. For example, S_i contains student names, identification numbers and first midterm grades taken by the students for a course. S_j contains student names, identification numbers and second midterm grades taken by the same students for the same course. We can define the class " Grade-up " as the members or instances of this class who have taken higher grade at the second midterm than at the first midterm.

4. Clustering

Clustering is a kind of classification where there is no supervisor that defines the classes. In clustering, data is organized as group of objects or as a hierarchy of groups. Since classes or clusters are not defined by a teacher or a supervisor, cluster

analysis is called “unsupervised learning”. Cluster analysis does not require any assumptions for inferring of the classes.

A cluster is a collection or group of similar objects. Everitt suggests the following definitions of a cluster.

1. *A cluster is a set of entities which are **alike**, and entities from different clusters are not alike.*

2. *A cluster is an aggregation of points in the test space such that the **distance** between any two points in the cluster is less than the distance between any point in the cluster and any point not in it.*

3. *Clusters may be described as connected regions of a multi-dimensional space containing a relatively **high density** of points, separated from other such regions by a region containing a relatively low density of points. (Jain, A.K. & Richard, C., 1988, p. 1)*

According to the first definition, we can group the objects or tuples in a database, such that objects have the same attribute-value. In second and third definitions, objects are assumed as points in the measurement space. When we look at these points in the plane, we can define the clusters.

When we look at the figure 3.3, we define four clusters with higher level of similarity. But at the local level of similarity, we can define nine clusters. This means that objects can be clustered with different purposes. So that, to give an operational definition of cluster is very difficult.

As it can be seen, different persons do not construct the same clusters from the same data. Criteria are subjective and depend on the educational and cultural background of person. Clustering techniques have an advantage, because they can apply objective criterion. When the data is huge, it takes very long time to cluster the objects with hand. But computer applications of clustering algorithms take very little time with respect to manual clustering.

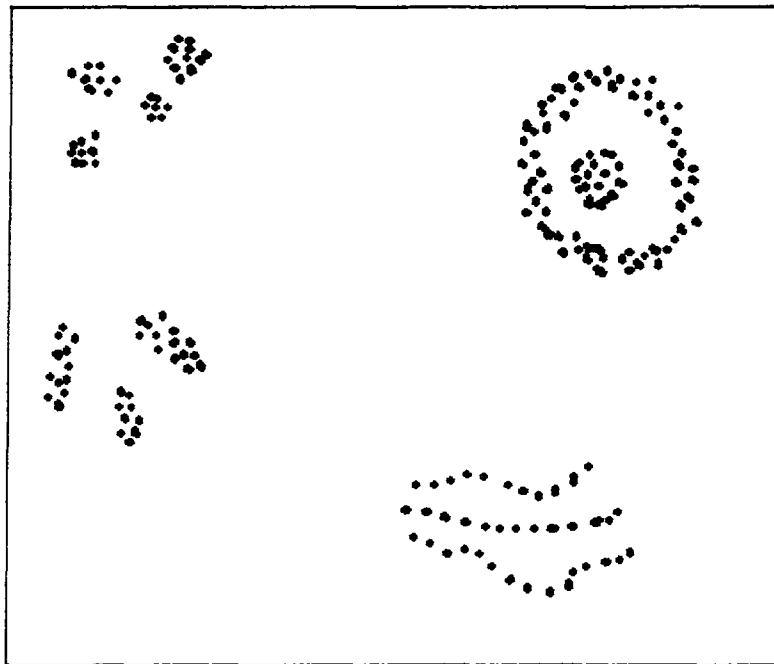


Figure 3.3 : Clusters of point pattern in two dimensions¹

Objects that cluster analysis uses, are called examples, individuals, subjects, cases, operational taxonomic units or tuples in different applications. The training set or a set of objects can be shown by two formats: a *pattern matrix* and a *proximity matrix*.

4.1. Pattern matrix:

Pattern matrix has n rows and d columns. Each row represents an object and each column represents a feature of objects. Each row is called a pattern. This means, each object is represented by a pattern. A pattern can be named d -place vector. We can say with respect to this definition of pattern matrix, any table of a relational database is a pattern matrix. For example, table 3.1 in page 16 can be an example of a pattern matrix. Table 3.1 has eight objects or individuals and each object has four measurements or features or attributes. We can say that table 3.1 is 8×4 pattern matrix.

¹ This figure is taken from Jain, Anil K. and Dubes, Richard C. , "Algorithms for clustering data", Prentice Hall, Inc., New Jersey, pp.2

We can show each as a point in a space which has d dimensions. d -dimensional space is called a *pattern space*.

4.2. Proximity matrix:

In clustering method, an index of proximity or likeness is used. Proximity index can be computed from a pattern matrix. A *proximity matrix* is a square matrix with $n \times n$ dimension where n is the number of the objects. Any element of proximity matrix at i th row and j th column is proximity index. This proximity index shows the degree of similarity or dissimilarity between i th object and j th object in pattern matrix. In proximity matrix, each row and column represents a pattern. The diagonal entries of a proximity matrix can be ignored, because, all patterns are assumed to have the same degree of proximity with themselves. The proximity matrix can also be assumed as a symmetric matrix. This means that all pairs of objects have the same proximity index and, the order of pairs of objects is not important.

A proximity index can show either a similarity or a dissimilarity. The meaning of large similarity index will be smaller dissimilarity index. For example, Euclidean distance between two patterns in a pattern space is a dissimilarity index but, the correlation coefficient is a similarity index.

A proximity index must satisfy the following properties where $d(i, j)$ denotes a proximity index between i th and j th patterns.

1. $\forall i, d(i, i) \geq \max d(i, j)$, for a similarity
2. $\forall i, d(i, i) = 0$, for a dissimilarity
3. $\forall (i, j), d(i, j) = d(j, i)$
4. $\forall (i, j), d(i, j) \geq 0$

There are several ways to determine a proximity index. Minkowski metric is one of them. Minkowski metric measures dissimilarity. In this method, i th row of the pattern matrix is denoted by the column vector x_i .

$$x_i = (x_{i1}, x_{i2}, \dots, x_{id})^T \quad \text{and} \quad i = 1, 2, 3, \dots, n$$

where d is the number of attributes, n is the number of objects or the number of the rows of pattern matrix and T denotes vector transpose. The Minkowski metric is defined by

$$d(i, j) = \left(\sum_{k=1}^d |x_{ik} - x_{jk}|^r \right)^{1/r} \quad \text{where } r \geq 1$$

Minkowski metric has additional two metric properties:

1. $d(i, j) = 0$ only if $x_i = x_j$
2. $d(i, j) \leq d(i, k) + d(k, j)$, for all (i, j, k)

When r is one, we can get the formula:

$$d(i, j) = \sum_{k=1}^d |x_{ik} - x_{jk}|$$

that is called Manhattan or Hamming distance. When r is two, we get the formula:

$$d(i, j) = \left(\sum_{k=1}^d |x_{ik} - x_{jk}|^2 \right)^{1/2} = [(x_i - x_j)^T (x_i - x_j)]^{1/2}$$

that is called Euclidean distance.

4.3. Clustering methods and algorithms

Clustering is a special kind of classification. The training set of clustering is a finite set of objects. Proximity matrix can be constructed by using relationships between objects. Rows and columns correspond to objects in proximity matrix. If we want to make meaningful cluster analysis, a meaningful proximity should be established between pairs of objects. Clustering algorithm accepts proximity matrix as one and the only input.

Classification can be divided into two parts; *non-exclusive classification* and *exclusive classification*. In exclusive classification, an object can be an instance of only one class or cluster. In non-exclusive classification, an object can be an instance of more than one class or cluster. For example, a student can belong to only one class, junior or senior. But, a student can take two or more courses. So that, the student may be in the list of students of different courses.

There are two types of exclusive classification; *extrinsic classification* or supervised learning and *intrinsic classification* or unsupervised learning. In extrinsic classification, a supervisor or a teacher defines the classes. In intrinsic classification, user constructs proximity matrix and classification algorithm which uses this proximity matrix.

There are two types of intrinsic classification method; *hierarchical clustering* and *partitional clustering*.

4.3.1. Hierarchical clustering:

In this method, a proximity matrix is transformed into a sequence of nested partitions. A hierarchical clustering is performed by a hierarchical clustering algorithm.

The objects in the training set are denoted by set X ,

$$X = \{ x_1, x_2, \dots, x_n \}$$

where n is the number of objects and x_i is the i th object. A partition, P , of set X divides X into subsets $\{ C_1, C_2, \dots, C_k \}$. C_1, C_2, \dots, C_k are called clusters and k is the number of clusters in partition P . The partition P satisfies the following properties:

$$\text{For } i \text{ and } j \text{ from } 1 \text{ to } k, i \neq j; C_i \cap C_j = \Phi$$

$$C_1 \cup C_2 \cup \dots \cup C_k = X$$

where “ \cap ” stands for set intersection and “ \cup ” stands for set union and Φ is the empty set. Let two different partitions be P_1 and P_2 . If every element of P_2 is a proper subset of an element of P_1 , it can be said that partition P_2 is nested into partition P_1 . We can construct partition P_1 by merging elements of P_2 . For example, let the P_1, P_2 and P_3 be

$$P_1 = \{ (x_1, x_2, x_3, x_4, x_5), (x_6, x_7), (x_8, x_9, x_{10}) \}$$

$$P_2 = \{ (x_1, x_3), (x_2, x_4), (x_5), (x_6, x_7), (x_8), (x_9, x_{10}) \}$$

$$P_3 = \{ (x_1, x_3, x_5, x_6), (x_2, x_4, x_7, x_8), (x_9, x_{10}) \}$$

P_1 has three clusters. P_2 has six clusters and $X = \{ x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} \}$. P_2 is nested into P_1 but P_1 and P_2 is not nested into P_3 and P_3 is not nested into P_1 and P_2 .

As an example, Hubert's algorithm is given below. Hubert's algorithm generates an *agglomerative*² hierarchical algorithm and uses *single-link* and *complete-link* methods. The input of this algorithm is proximity matrix. Clusterings are numbered $0, 1, 2, \dots, (n-1)$ and there are $n-m$ clusters in m th clustering C_m .

$$C_m = \{ c_{m1}, c_{m2}, \dots, c_{m(n-m)} \}$$

HUBERT'S ALGORITHM FOR SINGLE-LINK AND COMPLETE-LINK METHODS³

² An agglomerative algorithm start with the disjoint clustering. This algorithm places each of the n objects in an individual cluster and gradually merges these atomic clusters into larger and larger clusters until all objects are in a single cluster.

³ This algorithm is taken from Jain, Anil K. and Dubes, Richard C., "Algorithms for clustering data", Prentice Hall, Inc., New Jersey, pp.63

Step 1. Set $m \leftarrow 0$. Form the disjoint clustering with clustering number m .

$$C_0 = \{ (x_1), (x_2), \dots, (x_n) \}$$

Step 2a. To find the next clustering (with clustering number $m + 1$) by the single-link method, define the function Q_s for all pairs (r, s) of clusters in the current clustering as follows.

$$Q_s(r, s) = \min \{ d(i, j) \} : \text{the maximal subgraph of } G(d(i, j)) \text{ defined} \\ \text{by } c_{mr} \cup c_{ms} \text{ is connected}$$

Clusters c_{mp} and c_{mq} are merged to form the next clustering in the single-link hierarchy if

$$Q_s(p, q) = \min \{ Q_s(r, s) \}$$

$$\left[\min_{x_i \in c_{mp}, x_j \in c_{mq}} \{ d(i, j) \} = \min_{r \neq s} \left\{ \min_{x_i \in c_{mr}, x_j \in c_{ms}} \{ d(i, j) \} \right\} \right]$$

Step 2b. The function Q_c is used to find clustering number $m + 1$ by the complete-link method and is defined for all pairs (r, s) of clusters in the current clustering.

$$Q_c(r, s) = \min \{ d(i, j) \} : \text{the maximal subgraph of } G(d(i, j)) \text{ defined} \\ \text{by } c_{mr} \cup c_{ms} \text{ is complete}$$

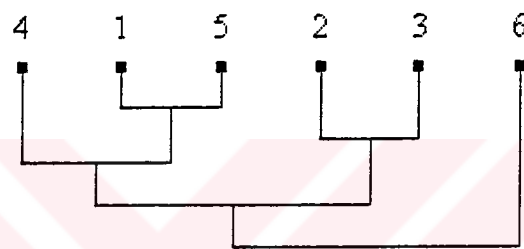
Cluster c_{mp} is merged with cluster c_{mq} under the complete-link method if

$$Q_c(p, q) = \min \{ Q_c(r, s) \}$$

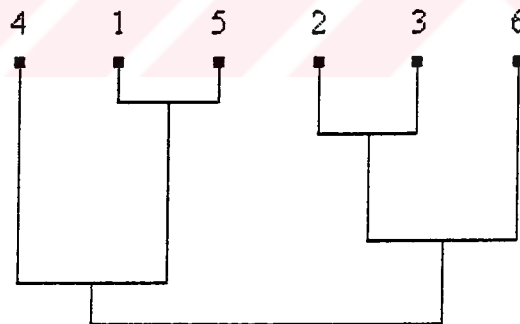
$$\left[\max_{x_i \in c_{mp}, x_j \in c_{mq}} \{ d(i, j) \} = \min_{r \neq s} \left\{ \max_{x_i \in c_{mr}, x_j \in c_{ms}} \{ d(i, j) \} \right\} \right]$$

Step 3. Set $m \leftarrow m + 1$ and repeat step 2. Continue until all objects are in a single cluster.

EXAMPLE: A proximity matrix for $n = 6$ is given as matrix D.

$$D = \begin{matrix} & \begin{matrix} X1 & X2 & X3 & X4 & X5 & X6 \end{matrix} \\ \begin{matrix} X1 \\ X2 \\ X3 \\ X4 \\ X5 \\ X6 \end{matrix} & \begin{bmatrix} 0 & 4 & 6 & 3 & 1 & 9 \\ 4 & 0 & 2 & 11 & 14 & 7 \\ 6 & 2 & 0 & 10 & 12 & 8 \\ 3 & 11 & 10 & 0 & 13 & 15 \\ 1 & 14 & 12 & 13 & 0 & 5 \\ 9 & 7 & 8 & 15 & 5 & 0 \end{bmatrix} \end{matrix}$$


Dendrogram for single-link hierarchical clustering.



Dendrogram for complete-link hierarchical clustering.

A dendrogram adds a the visual impact to hierarchical clustering methods. A dendrogram is a special kind of tree structure. A dendrogram consist of nodes and lines that connect nodes. Connected nodes represent clusters. These clusters are nested into

one another. Clustering can be created by cutting a dendrogram horizontally at any level of proximity.

4.3.2. Partitional clustering:

Partitional clustering methods are the non-hierarchical clustering methods. Partitional clustering methods generate a single partition of data by retrieving natural groups present in the data. Both clustering methods, hierarchical and partitional clustering methods, use their appropriate domains of applications. Hierarchical clustering methods require only the proximity matrix generated from the pattern matrix of the objects. Partitional clustering methods generally require the pattern matrix itself. Generally assumed that the features of the object in pattern matrix have been measured on a ratio scale.

The application domains of hierarchical clustering methods are generally biological, social, and behavioral sciences. Because we need to construct taxonomies at these search areas. The application domain of partitional clustering methods are engineering sciences. Because the single partitions are important in engineering applications. Partitional techniques are used for large databases, but hierarchical techniques are not. Because the dendrograms are practical for a few hundred patterns.

In partitional clustering, n patterns in a d -dimensional metric space is given. Clustering algorithm should determine a partition of the patterns into K clusters. In a cluster, the patterns are more similar than the patterns in different clusters. An appropriate clustering criterion must be chosen. Criteria can be named as *global* or *local*. A prototype is suggested to each cluster as the global criterion. Patterns that has high resemblance to that prototype belong to that cluster. Local structure in the data is utilized to form clusters as the local criterion.

There are two difficulties in partitional clustering problem. The first difficulty is to translate the selected criterion into a mathematical formula. Criteria is dependent on the parameters of problems. Criteria must be simple for computation. At the same

time, criteria must be complex enough to represent various data structures. The second difficulty is that the number of partitions is very big in partitional clustering methods. The number of clustering can be computed with the formula

$$S(n, K) = \frac{1}{K!} \sum_{i=1}^K (-1)^{K-i} \binom{K}{i} i^n$$

where n is the number of patterns, K is the number of clusters and $S(n, K)$ is the number of clustering of n objects into K clusters. The boundary conditions on this equations are

$$S(n, 1) = 1, \quad S(n, n) = 1, \quad S(n, K) = 0 \text{ if } K > n$$

The most commonly used partitional clustering methods are *square-error* clustering methods, *nearest-neighbor* clustering methods and *fuzzy* clustering methods.

Square-error clustering methods use the square-error clustering criteria. The square-error criteria are used to obtain fixed number of clusters with minimum square-error. Minimizing square-error is equivalent to maximizing the between-cluster variation.

Suppose a pattern matrix with $n \times d$ dimensions is given and this pattern matrix has been partitioned into K clusters $\{c_1, c_2, \dots, c_K\}$. n_i is the number of patterns into the cluster c_i , so that

$$\sum_{i=1}^K n_i = n$$

$$m^{(i)} = \frac{1}{n_i} \sum_{j=1}^{n_i} x_j^{(i)}$$

The mean vector $m^{(i)}$ of cluster c_i is the centroid of the cluster, where $x_j^{(i)}$ is the j th pattern into the cluster c_i . The square-error e_k^2 for cluster c_k can be computed from the formula

$$e_k^2 = \sum_{i=1}^{n_k} (\mathbf{x}_i^{(k)} - \mathbf{m}^{(k)})^T (\mathbf{x}_i^{(k)} - \mathbf{m}^{(k)})$$

This square-error is also called the within-cluster variation. The square-error for the present clustering containing K clusters

$$E_K^2 = \sum_{k=1}^K e_k^2$$

The problem of a square-error clustering method is to minimize E_K^2 for fixed K.

An iterative partitional clustering algorithm is given below¹.

ALGORITHM FOR ITERATIVE PARTITIONAL CLUSTERING

Step 1. Select an initial partition with K clusters.

Repeat step 2 through 5 until the cluster membership stabilizes.

Step 2. Generate a new partition by assigning each pattern to its closest cluster center.

Step 3. Compute new cluster centers as the centroid of the cluster.

Step 4. Repeat step 2 and 3 until an optimum value of the criterion function is found.

Step 5. Adjust the number of clusters by merging and splitting existing cluster or by removing small, or outlier, clusters.

In nearest-neighbor clustering methods, two patterns should be considered similar if they have the same neighbors. These two patterns should belong to the same cluster. The user decides a threshold on the nearest-neighbor distance. A simple

¹ This algorithm is taken from Jain, Anil K. and Dubes, Richard C. , "Algorithms for clustering data", Prentice Hall, Inc., New Jersey, pp.96

clustering algorithm for nearest-neighbor clustering method is given below where x_i is the i th pattern of the pattern matrix⁵.

NEAREST-NEIGHBOR CLUSTERING ALGORITHM

Step 1. Set $i \leftarrow 1$ and $k \leftarrow 1$. Assign pattern x_1 to cluster C_1 .

Step 2. Set $i \leftarrow i + 1$. Find the nearest neighbor of x_i among the patterns already assigned to clusters. Let d_m denote the distance from x_i to its nearest neighbor in cluster m .

Step 3. If $d_m \leq t$, then assign x_i to C_m . Otherwise, set $k \leftarrow k + 1$ and assign x_i to a new cluster C_k .

Step 4. If every pattern has been assigned to a cluster, stop. Else, go to step 2.

Fuzzy clustering methods are used if the clusters are touching or overlapping. Some patterns may seem to belong to two clusters. Such patterns belong to a cluster with a grade of membership. The value of grade of membership must be in interval $[0, 1]$. If a pattern x_i belongs to the q th cluster then the membership grade for pattern x_i in a q th cluster, $f_q(x_i)$, is 1. If the pattern x_i does not belong to the q th cluster then $f_q(x_i)$ is 0. These clusters are called *crisp* clusters. With fuzzy clusters, the membership grade must satisfy the following two conditions:

$$1 \geq f_q(x_i) \geq 0 \quad \text{and} \quad \sum_q f_q(x_i) = 1$$

5. Rules

After the classes are found either with supervised or with unsupervised learning, the system should construct the rules for these classes. This means that the system should find descriptions for each class. These descriptions can only be handled from

⁵ This algorithm is taken from Jain, Anil K. and Dubes, Richard C., "Algorithms for clustering data", Prentice Hall, Inc., New Jersey, pp.128

predicting attributes of the training set S . If tuples or objects that satisfy the description, are positive examples, others that do not satisfy the description, are negative examples.

Descriptions are a group of conditions on attributes, because attributes of objects are only stored in training set and there is not any relationship among objects.

Many data mining tools use relational algebra to construct a subset of the selected conditions as a description. We can formulate the description such that:

A is the set of predicting attributes and $\text{Dom}(A_i)$ is the domain of A_i . An *elementary description* is

$$A_1 = c_1 \wedge A_2 = c_2 \wedge \dots \wedge A_n = c_n \quad \text{and} \quad A_i \in A$$

$$i \neq j \quad \text{then} \quad A_i \neq A_j \quad \text{and} \quad c_i \in \text{Dom}(A_i)$$

A *description* or pattern is a non-empty disjunction of elementary descriptions. $A_i = c_i$ is called an *attribute-value condition*. The *description space* D is the set of all possible descriptions.

Some data mining system only find elementary descriptions but some data mining systems can find all possible descriptions. Number of descriptions can be counted because we assume that domain is finite. Let $\text{Dom}(A_i)$ be domain of any attribute A_i , n be the number of attribute. We can count the number of elementary descriptions with the formula:

$$\prod_{i=1}^n (|\text{Dom}(A_i)| + 1)$$

The total number of descriptions can be counted by the formula:

$$2^{\prod_{i=1}^n (|\text{Dom}(A_i)| + 1)} - 1$$

Set of descriptions can be formulated as:

$$A_1 \in \{ c_{1,1}, c_{1,2}, \dots, c_{1,m_1} \} \wedge A_2 \in \{ c_{2,1}, c_{2,2}, \dots, c_{2,m_2} \} \wedge A_n \in \{ c_{n,1}, c_{n,2}, \dots, c_{n,m_n} \} = (A_1 = c_{1,1} \wedge A_2 = c_{2,1} \wedge \dots \wedge A_n = c_{n,1}) \vee (A_1 = c_{1,2} \wedge A_2 = c_{2,2} \wedge \dots \wedge A_n = c_{n,2}) \vee \dots \vee (A_1 = c_{1,m_1} \wedge A_2 = c_{2,m_2} \wedge \dots \wedge A_n = c_{n,m_n})$$

As a result, we can suggest a definition for the rule that is below:

‘ ‘ If D then C ’ means that if an object satisfies D then this object belongs to C where D represents definition and C represents class. ’

EXAMPLE : In this example, we will use table 3.1 in page 15 as a database. Three classes can be defined: ‘ Bird ’, ‘ Fish ’, and ‘ Others ’. The system can construct the following three classification rules:

If Organ = Wings and Ismammal = No then Bird

If Organ = Fins and Ismammal = No then Fish

If (Organ \neq Wings or Ismammal \neq No) and

(Organ \neq Fins or Ismammal \neq No) then Others

Positive examples of the class ‘ Bird ’ are stork, hawk, ostrich and penguin. Positive examples of the class ‘ Fish ’ are shark and salmon. And finally, positive examples of the class ‘ Others ’ are bat and sea turtle.

Many different descriptions can be constructed. Each description may construct classes with different instances. Unfortunately, we can not know, which descriptions will correctly classify unseen examples. For example, if we choose the classification rule as ‘ If Organ = Wings then Bird ’, bat will belong to the class ‘ Bird ’ but, bat is not a bird. And according to the given rule ‘ If Organ = Fins and Ismammal = No then Fish ’, dolphin will not belong to the class ‘ Fish ’, because dolphin is mammal. As it can be seen that all knowledge inferred from observation is not valid.

CHAPTER FOUR

SEARCH ALGORITHMS

As we see before, data mining system constructs many classes and descriptions that describe these classes. Some of these descriptions can classify unseen examples correctly than others and can describe relationships between objects in the data used. The problem is to find the best descriptions among the possible descriptions set D constructed by the data mining system. This problem can be named as *search problem*.

Data mining systems has a quality function to measure the quality of a description. These systems initially choose a description, initial description, and iteratively modify the initial description by using quality function. Thus, data mining systems try to improve the quality of description and to get best description. Both, the set of descriptions and the quality function together, is called *search space*.

1. Search space

We can define the search space as $(\mathcal{D}, f, \mathcal{O})$ where \mathcal{D} is a set of descriptions, f is a quality function and \mathcal{O} is a set operations on descriptions in the set \mathcal{D} .

1.1. Description space

The description space \mathcal{D} is the set of all possible descriptions constructed by the data mining system. A subset of training set S that defined by each description D in \mathcal{D} , is called *cover* $\sigma_D(S)$.

1.2. Operations

There are two types of operation: *Generalization* and *specialization*.

Generalization:

If we apply a generalization operation to a description D in \mathcal{D} , we get a new description D' . D' contains more objects than the description D . If an object belongs to D , it also belongs to D' . But any object of D' may not belong to D .

So, we can say that $\sigma_D(S) \subseteq \sigma_{D'}(S)$. A rule can be correct or in other words, a rule can classify the objects correctly, but a generalization of this rule may not be correct. This means that the generalization operation is not truth preserving. But generalization rules are falsity preserving. If an object is covered by D , but it is not an example of class C , (i.e. the object is not correctly classified by the rule), then the object falsifies the rule. If an object falsifies any rule, then it will also falsify any of generalizations of this rule.

Dropping condition operation is a special case of generalization operation. With dropping condition, the set S_i that $S_i \subseteq S$, is extended to the entire domain Dom_i . *Climbing generalization tree* operation is also a special case of generalization operation. With climbing generalization tree operation, we construct a compact representation of the class.

Specialization:

If we apply a specialization operation to description D in \mathcal{D} , we get a new description D' . D' contains less objects from the description D . If an object belongs to D' , it also belongs to D . But any object of D may not belong to D' . So, we can say that $\sigma_{D'}(S) \subseteq \sigma_D(S)$. As it can be seen that, the specialization operation is the inverse of the generalization operation.

1.3. Domains of the attributes

User should define the structure of the domain of the attributes in the database which generalization and specialization operation will be applied on. There are three basic types of structure of the domain.

Nominal (Categorical):

In this type of structure, symbols or names in the domain are independent and the values of an attribute are not ordered.

Linear:

In this type of structure, domain is totally ordered. Linear domain can be *ordinal*. For example, values of the domain of an attribute can be low, medium or high. We can not use the mathematical operations such as summation or multiplication. The linear domain can be an *interval* domain such that we can apply summation operation on the elements of domain, but we cannot apply multiplication. The linear domain can be *ratio* domain. We can apply both summation and multiplication on the elements of the domain.

Partially ordered:

The domain is partially ordered. Partially ordered domain has a hierarchical form, where a parent node represents a more general concept than its children. Any symbol is smaller than the top symbol.

1.4. Quality function

The quality function produces a numeric values. Each value belongs to a specific description and indicates the quality of the description. A description should classify any new, unseen object correctly. This statement means that a description should be *valid* generally. And, in unsupervised learning, a description should be *correct* with respect to defined classes. So that, a description has two criteria, validity and

correctness. We can assign a value to each criterion and these criteria can be combined with using a function to compute the quality of the descriptions.

Validity:

It is accepted that the number of objects is limited in a database. But this cannot be seen in real world. So that, the correctness of a rule cannot be verified for all possible situations. This means that we cannot prove the validity of a rule, in general. Most data mine systems rely on that the simpler description describes relationships between objects, approximately best. This rule is known as Ockham's razor. The quality function for validity, f_v , has higher value for simpler descriptions.

Correctness in supervised learning:

If all positive examples belong to class C and any negative example does not belong to class C with respect to description D, it can be said that description D is correct. In other words, if $\sigma_D(S) = C$ then the description D is correct.

Two probabilistic concepts are used to determine a rule or a description is correct; classification accuracy and coverage. If a rule is not correct it may be complete or deterministic. These concepts are explained below.

The classification accuracy is the relative portion of the number of elements of the training set S which are covered by the description D that is also covered by the class C:

$$\text{classification accuracy} = \frac{|\sigma_D(S) \cap C|}{|\sigma_D(S)|}$$

The value of classification accuracy is an element of the interval [0,1]. The classification accuracy is the probability that an object covered by the description D belongs to the class C.

The coverage is the relative portion of the number of elements of class C which are also elements of the training set S covered by the description D:

$$\text{coverage} = \frac{|\sigma_D(S) \cap C|}{|C|}$$

The value of coverage is also an element of the interval [0,1]. The coverage is the probability that an object belongs to the class C is also covered by the description D.

If the coverage is equal to 1, the description is a necessary condition for the class. In this situation, any object belonging to the class is also covered by the description. The class C is a subset of $\sigma_D(S)$, $C \subseteq \sigma_D(S)$. And the rule is called as *complete rule*.

If the classification accuracy is 1, the description is a sufficient condition for the class. In this situation, any object covered by the description belongs to the class. $\sigma_D(S)$ is a subset of class C, $\sigma_D(S) \subseteq C$. And the rule is called as *deterministic rule*.

If the classification accuracy and coverage are equal to 1, the description is both a necessary and sufficient condition for the class. In this situation the class and the set $\sigma_D(S)$ are the same. And the rule is called as *correct rule*.

The quality function for correctness, f_c , has a value which belongs to the interval [0,1]. When the description is correct, the value of f_c is 1. When the description is incorrect, the value of f_c is smaller than 1. G. Piatetsky-Shapiro and W. J. Frawley proposes some principles for the construction of the correctness criterion f_c such that:

- 1- If the classification accuracy is equal to the probability that any object in the training set S belongs to the class C then the description D and class C are statistically independent. The value of f_c is 0 and the description is wrong.

$$\frac{|\sigma_D(S) \cap C|}{|\sigma_D(S)|} = \frac{|C|}{|S|}$$

- 2- f_c monotonically increases with $|\sigma_D(S) \cap C|$ when $\sigma_D(S)$ and C remain the same.
- 3- f_c monotonically decreases with $|\sigma_D(S)|$ or $|C|$ when $\sigma_D(S) \cap C$ remains the same.

Correctness in unsupervised learning:

If we use the correctness criteria explained above, any simple description that covers all examples in the training set will have high quality. For example, the description true will cover all tuples in the training set and assigns all objects to their correct and only one class. because the description is extremely simple.

This shows that a quality function should not only depend on the simplicity of the description. Michalski and Stepp¹ suggest that a quality function should also depend on *fit*: how close does the description approximate the set of examples. For the property fit, we should compute the ratio

$$\frac{|\sigma_D(S)|}{|\sigma_D(U)|}$$

where $|\sigma_D(S)|$ is the number of objects in the training set S covered by the description D and $|\sigma_D(U)|$ is the number of the objects in the universe covered by the same description D . If the description covers only objects in S and does not cover objects that do not belong to S , this ratio is exactly one.

Combining criteria:

The criteria validity f_v and correctness f_c denote the quality of a description. In some problems, many other criteria such as the cost of evaluating the description or the cost of measuring attributes, could be taken into account. For the overall quality, we should combine these criteria.

In general, there are two ways to compute the overall quality. In first way, a weight to each criterion should be assigned and weighted sum of the qualities for

these criteria gives the overall quality. Let f_1, f_2, \dots, f_n are the values of criteria and w_1, w_2, \dots, w_n are the weights of these criteria respectively,

$$\text{Overall quality} = f_1w_1 + f_2w_2 + \dots + f_nw_n$$

Second way of the computation of the overall quality is called *lexicographic evaluation functional* (LEF)². In this way, the criteria f_1, f_2, \dots, f_n are ordered and t_1, t_2, \dots, t_n are tolerances or thresholds of these criteria respectively. LEF of these criteria can be shown as.

$$\text{LEF} = ((f_1, t_1), (f_2, t_2), \dots, (f_n, t_n))$$

The LEF determines the most suitable description from the given set of descriptions in this way: at first, all descriptions are ordered based on the value of the first criterion. Only the descriptions within the range defined by the threshold t_1 from the best description are chosen. Best description in the process is top of the ordered list. Chosen descriptions are ordered based on the value of the second criterion and the best are retained. When this process are applied to the last criterion, the best description is handled.

2. Search algorithm

There are many different algorithms to define descriptions and set the classes. Generally, basic idea of these algorithms is to start with an initial description. Algorithm iteratively modifies the initial description with applying some operations until the quality of description exceeds user-defined threshold.

2.1. Initial description

There are two approaches, *bottom-up* and *top-down*. The choice of the initial description and applied operations are different in these two approaches.

¹ Michalski, R.S. & Stepp, R.E. Learning from observation: conceptual clustering. *Machine Learning, an Artificial Intelligence Approach*, volume 1 California 1983 pp: 331-363

² Michalski, R.S. , Carbonell, J.G. & Mitchell, T.M. *Machine Learning, an Artificial Intelligence Approach*, volume 2. California, 1986. pp:83-134

Bottom-up (or data-driven):

At first, a target class and its all examples are defined. The initial description is the set of examples of the target class. But generally, this description is correct and too complex. Thus, the description should be modified iteratively to reduce the complexity. After the modification operation, more general rule can be handled that classifies the objects correctly within a certain tolerance. Dblearn system is an example of this technique.

Top-down (or model-driven):

At first, we choose an initial description. For example, this description can cover the universe. Then, we apply a sequence of generalization and specialization operations. The initial description will change with these sequence of operations. When the description quality exceeds the threshold, we stop. BACON system, ID3-system and META-DENDRAL system are the examples of the top-down technique.

2.2. Search graph

Let \mathcal{O} be a set of operations $\mathcal{O} = \{ o_1, o_2, o_3, \dots, o_n \}$ and the initial description be D_1 . When any of the operations o_i is applied on D_1 , a new description D_{2i} is produced. All operations cause n new descriptions $D_{21}, D_{22}, \dots, D_{2n}$. A *search graph* can be constructed by repeatedly applying all operations on the new descriptions. This search graph is illustrated in Figure 4.1.

In figure 4.1, the nodes denote the descriptions and the arcs denote the operations. This graph may not be a tree. Because, different sequence of operations may produce the same description.

Operations are applied on the descriptions until some sequence of operations is found that constructs a description or a rule which its quality exceeds the threshold.

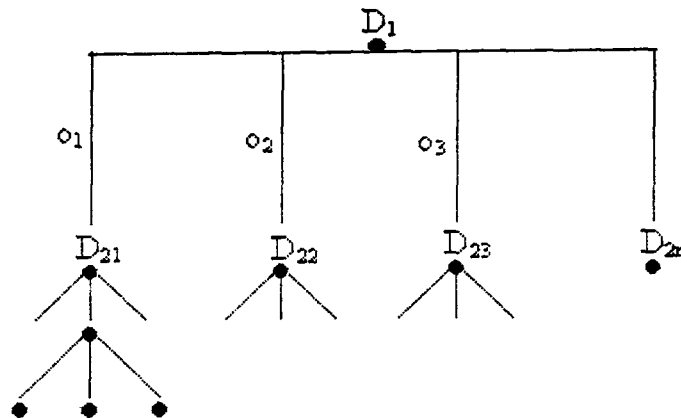


Figure 4.1: A search graph

Search strategy is the way to get these sequence of operations. There are mainly two search strategy: Irrevocable search and tentative search.

Irrevocable search:

In this search strategy, we select an operation and apply. We do not make backtracking for reconsideration later. We can evaluate a single path in the graph.

Tentative search:

In this search strategy, we select an operation and apply. But we can return later to this point to apply some other operation. There are mainly two types of the tentative search strategy; *backtracking* and *graph search strategies*. In backtracking, we examine the nodes in a dept-first manner. A single path is examined. At the end of the path, if the system can not construct a description which its quality does not exceed the threshold, backtracking is performed. In graph search, we examine the nodes in a breadth-first manner. All new nodes are equal distance from the starting node. In graph search, we need more storage.

All these search strategies are not uniform. We choose the operations arbitrarily and we do not use any heuristic information. The system generates many descriptions. The quality of each description should be computed. The computation of description quality in a database is very expensive.

3. Heuristic search

The aim of the heuristic search is to find the shortest path to a goal node that, constructed description has sufficient quality. This path is unknown at the beginning. The heuristic strategy requires information about the search domain for selecting suitable operations. This information is called heuristic or domain knowledge.

3.1. Hill climber

One of the heuristic search is the hill climber strategy. In the hill climber strategy, the operation that results in the greatest quality is chosen. The system should compute the quality of all possible extensions. If the computation of quality is expensive, the system may estimate the quality. User should define an *expectation function*. Expectation function returns the quality of the generated description by applying a particular operation.

3.2. Limitations on the operations

Heuristic knowledge is specific to a part of the domain. This information has to be supplied by the user. There are two forms of heuristic knowledge.

Irrelevant attributes:

Some attributes in the database can be chosen as the irrelevant attributes. For example, first name of the student is not important in the question “ How many students have finished the Law Faculty?”. So, the user could define the relevant and the irrelevant attributes for the classification. By using this way, the number of descriptions can be reduced.

Some attributes depend on the value of other attributes. For example, military knowledge is necessary if the person is male. So that, the attribute “military” can not be regarded for some classes.

Interrelationships between attributes:

Some attributes value can be computed from other values of attributes. For example, the volume of a cube can be computed from length of the one edge of the cube. So that, the quality of a description will not increase when adding the condition 'volume'.

The heuristic knowledge can also be the previous knowledge of the user or previously constructed rules and classes. Some information may not be coded as heuristic knowledge in the database, but this information can be known by the user, and the user uses this information in the search process. Also, previously discovered rules and classes can be used for further investigation by the system. This process is important when the set of examples is updated.



CHAPTER FIVE

PROBLEMS

In data mining process, we accept that descriptions or classification rules exist in the data set. This may be true for some artificial data sets which are used in machine learning. But it is not always true when databases are used. We come face to face with several problems when the training set is a database. These problems can be limited of the supplied information, missing data, the size of the database and the problems from dynamic behaviors of the database.

1. Limited information

1.1. Incomplete information

In supervised learning, we choose predicted attributes to determine the classes. These attributes are relevant for classification. But some of these predicted attributes may not be recorded in the database. And to construct a rule for classification may not always be possible by using known predicting variables.

There are two approaches when we have unknown predicting variables: we can either restrict ourselves with known variables. By which, we can construct only deterministic rules, and cannot find some valuable information that is hidden in the database.

Or we can search rules that are not necessary to determine the classes correctly, because it is possible that an object, covered by the description, belongs to a class.

Such rules can be called as probabilistic rules. We can obtain very important information about relationships between objects in the database. For example, smoking is not necessary and not sufficient condition for cancer. But, still, this relationship is considered very important.

1.2. Sparse data

A classification rule constructed by the data mine system has to set the class boundaries. We can investigate the quality of these boundaries if the database contains examples which are just within (near misses) or outside (near hits) the class. This means that a database must have facts which represent all possible behavior. But in real world, facts in a database represent only a small subset of all possible behavior. So that class boundaries can be incorrect or vague.

For a solution, we need additional information. The system might search additional information in the database for interesting examples.

2. Data corruption

We assume that all examples in the data set have correct values. An object in the database has many properties or attributes. Some attributes may have values which are based on measurement or subjective judgments which may cause some errors in the value of attributes. Such errors are called noise. And also, some attribute values may be missing. Both causes misclassification.

2.1. Noise

We meet problems caused by noise, when the system construct the descriptions and classify the examples by using these descriptions.

Constructing descriptions:

In a noisy training set, constructed descriptions may cover corrupted examples. Therefore, the system should decide whether an example is corrupted or not. Corrupted examples should be ignored.

Classifying examples:

Previously constructed descriptions taken from a training set can be used to classify the previously unseen examples. Corrupted examples may cause misclassification. Of course, misclassification of unseen examples is expected at a low level. For the solution, we can compare the rules constructed from the noisy training set with the rules constructed from the same but noise free training set. If there is a small amount of this misclassification, the rules constructed from the noisy training set can be used in practice.

2.2. Missing attribute values

We meet two problems which missing attribute values cause, at two different levels of learning process.

Constructing descriptions:

The system may not take into consideration the examples with missing attributes. Or, the system can replace the missing value with the new approximate value. This value can be computed from the value of other attributes by the statistical methods. Or, simply, missing values are filled with the value ' unknown ' and these values can be used in the descriptions.

Classifying examples:

Unseen examples with missing attribute values can be classified by previously constructed descriptions. If these descriptions contain conditions on some of these attributes, they cannot be applied.

3. Databases

Database is a training set used in data mining. This training set has some difference from the training set used in machine learning. The training set used in machine learning is constructed by the user for a special purpose.

3.1. Size of database

In machine learning, the training set is small, (for example, the training set which contains thousand objects. is considered to be a large training set) but on other hand the number of objects in a database and the amount of properties per objects are generally very large.

Information per object:

Most databases contain many attributes. For example in the database for students in Law Faculty, the number of attributes is approximately 200. In reality, much information per objects is an advantage. With more information per object, we can learn true relationships between objects, but the number of constructable description increases with the number of information. The number of description depends on the size of domain of attributes. Roughly, the number of constructable descriptions is 2^λ , where λ is the sum of the size of domain of attributes. To overcome this problem. we eliminate some attributes which are considered not necessary.

Number of objects:

The problem is faced when we try to verify the quality of each constructed description. We use statistical tests in this verifying process. This test needs information about the number of examples covered by the description, or the distribution of values in this set. This test is very expensive in huge databases. We can use two techniques to overcome this problem.

1. Multiple descriptions:

In a single iteration of the search process, multiple descriptions can be constructed. We can compute their quality simultaneously by a single but complex database access.

2. Windows:

We choose a subset of database as a representative sample. This sample is called a window. This sample is small with respect to the entire database. It contains a few thousand objects. We can compute the quality of a description using this window. Then, the best descriptions are tested on the real databases.

Of course, the actual probability of the rules may not be equal to the predicted probability. We choose some incorrectly classified examples. Then, we add these incorrectly classified examples to the window and modify the rules using this new window. This modification process is called *incremental learning*.

3.2. Updates

In the course of time, the database will change. Some properties of examples can change, some examples can be added or removed. Because of this reason, the quality of some rules will decrease. When we run these rules, some objects are classified incorrectly. The system should adapt to such changes, and rules should be adjusted.

When too many incorrect predictions are made, data mine systems start the process of rule adjustment. Some kinds of incremental learning can be used to overcome this problem. A kind of incremental learning is *learning with full memory*. In this type of learning, the system remembers all examples. Other kind of incremental learning is *learning with partial memory* which is opposite of first type of learning. It is obvious that new rules are guaranteed to be correct with respect to all old and new training examples.

CHAPTER SIX

KNOWLEDGE REPRESENTATION

In this chapter, we will discuss some kind of knowledge representations. In previous chapters, we used relational algebra (or selection conditions) to present the condition in supervised learning and to describe the database in unsupervised learning.

The other representation methods are First Order Logic (FOL), propositional representation, structured representation and neural networks.

1. Propositional-like representations

In propositional representations, we use logic operators to formulate the descriptions. These descriptions consist of the values of attributes. '(high school=Normal \vee high school=Anadolu) \wedge father's education=University' is an example. This formula is in Conjunctive Normal Form (CNF), conjunction of clauses, where clauses are disjunctions of attribute value conditions. We can re-write this formula as the set-description 'high school \in { Normal, Anadolu } \wedge father's education \in {University }'.

An alternative form of CNF is Disjunctive Normal Form (DNF), disjunction of clauses, where clauses are conjunctions of attribute value conditions. Previous studies indicated that, the generated descriptions with CNF representations are smaller than the DNF representations.

The above examples are not really propositional, because, in propositional representation, we must use variables. This is the reason why we call them propositional –like.

1.1. Decision trees

With a decision tree, the examples are classified to a finite number of classes. In the tree, nodes are labeled with attribute names, the edges are labeled with possible values for this attribute, and the leaves are labeled with the different classes. An object is classified from the bottom of the tree, by taking the values of the attributes written on the edges.

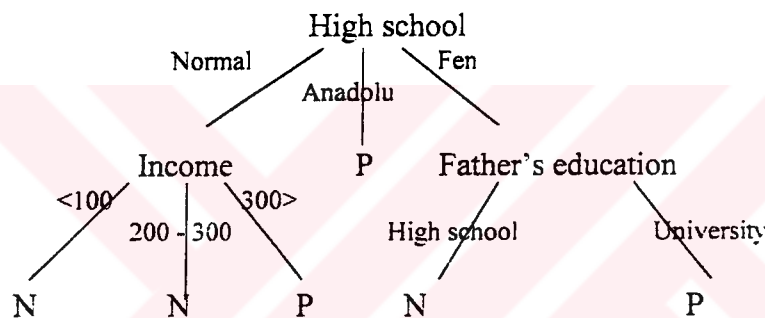


Figure 6.1: An example for the decision tree

In figure 6.1, 'high school' is an attribute name and, 'Normal', 'Anadolu' and 'Fen' are the possible values of the attribute 'high school'. P represents positive examples and N represents negative examples.

Decision tree is suitable for supervised machine learning systems. But, for realistic application, the decision tree becomes very large. There has been some research to transform the decision tree into other representations.

1.2. Production rules

Production rules are a transformation of the decision tree and a propositional-like representation. In expert systems, production rules are widely used for representing

knowledge. Production rules can easily be interpreted, because a single rule can be understood without reference to the other rules.

As an example, we can transform the decision tree in the figure 6.1 to the propositional-like production rules. We will use the conjunctive normal form.

If high school=normal and income < 100 then class = N

If high school = Fen and father's education = University then class = P

If high school = Anadolu the class = P

1.3. Decision list

Decision list is another propositional-like representation. Any knowledge structure constructed as a decision list representation can be transformed to a decision list or DNF representation or CNF representation. A decision list is a list of pairs which first item of a pair is an elementary description ϕ_i and the second item is C_i .

$$(\phi_1, C_1), (\phi_2, C_2), (\phi_3, C_3), \dots, (\phi_k, C_k)$$

The last description has a constant value; true. If the last index of a description is j the ϕ_j covers object \bullet and \bullet belongs to class C_j . A decision list can be extended as a rule 'if ϕ_1 , then C_1 , else if ϕ_2 then C_2 ,else C_r '.

1.4. Ripple-down rule sets

Sometimes, we need to represent some exceptions in the rules, such that 'if ϕ_i then C_i unless ϕ_j '. We can add an exception rule 'if ϕ_j then C_j '. But, any object for which ϕ_j is true will be assigned to class C_j globally, whereas we want the exception to be local to the rule 'if ϕ_i then C_i '. As a result, the decision list will be difficult to understand.

Ripple-down rule sets represent exceptions in a more localized manner. These rules consist of conditions and exceptions to these conditions that are local to the rule. These rules are nested if-then statements such that;

```

if  $\emptyset_i$  then
    if  $\emptyset_j$  then  $C_j$ 
        else  $C_i$ 
    else  $C_j$ 

```

Above example is a ripple-down rule set with depth 2. Here, we do not need a global ordering of the rules. as in the decision lists.

2. First order logic

The propositional-like representation has some disadvantages. We cannot represent the patterns in terms of relationships among objects or attributes. For example, we can not construct a class which contains students that take same grade from 'Anayasa' and 'Hukuk Baslangici'. We need more powerful representation to state that any student where 'Anayasa = Hukuk Baslangici' belongs to the class.

In the learning process with the propositional-like representation, it is difficult to incorporate domain knowledge. It is accepted that domain knowledge consists of constraints on the descriptions, generated by the system. But, the domain knowledge is rarely complete and consistent.

Learning systems construct descriptions within the limits of a fixed vocabulary of propositional attributes. We can increase the set of patterns and the comprehensibility of the representation by using auxiliary predicates.

We need a more powerful representation to overcome these problems. Some kind of First Order Logic is used to represent knowledge. This type of representation is called ***Inductive Logic Programming***. The aim of the inductive logic programming is to construct a First Order Logic program. This program has the training set as its logical consequence.

When we find complex descriptions in a less powerful representation, First Order Logic allows us to find simple descriptions for classes. As a result the computational complexity of construction of description decreases. The set of possible descriptions gets larger. Larger set of descriptions may make learning easier. It may be easier for

the learning algorithm to produce a nearly correct answer from a rich set of alternatives than from a small set of alternatives. But, it is difficult to select the best description. A solution is to search for particular descriptions only.

3. Structured representations

There are mainly two types of structured representation; semantic nets and frames. These representations are not more powerful than First Order Logic, but they provide a more comprehensible representation. We can state subtype relationships among objects by structured representations. Structured representations can be expressed as a First Order Logic program.

3.1. Semantic nets

A semantic network is a directed graph. The nodes of this graph denote concepts and the arcs denote relationships between those concepts.

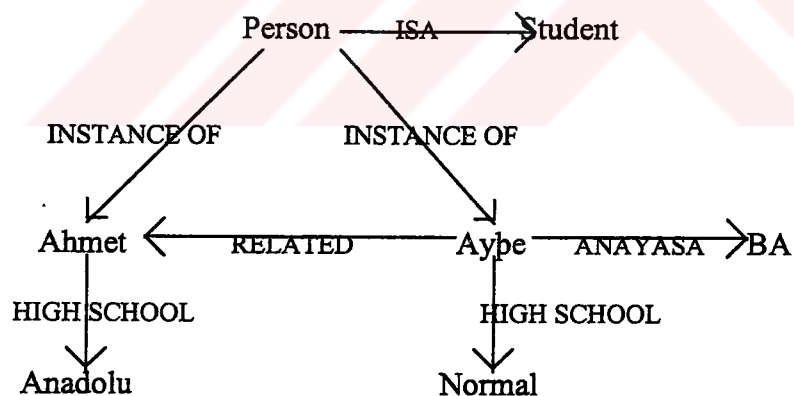


Figure 6.2: An example of semantic nets

In the figure 6.2, we can see an example of the semantic network. In this example, 'Related', 'High school', 'Anayasa' are the relationships between concepts and, 'instance-of' and 'isa' are the relationships between concepts and classes. If we

represent these relationships with a different network, they can be more comprehensible.

As we stated before, we can express this semantic network in First Order Logic. Each arc can be expressed as a binary predicate and nodes can be expressed as terms:

High school (Ahmet, Anadolu).

High school (Ayse, Normal).

Related (Ahmet, Ayse).

Anayasa (Ayse, BA),

Person (Ahmet).

Person (Ayse).

$\forall x. \text{person}(x) \rightarrow \text{Student}(x)$.

With semantic nets representation, we can find all information related to a particular object. Each example or object is a semantic net for data mining. We use graph-manipulations to find patterns. These patterns are only subgraphs which are shared by all objects of the same class.

3.2. Frames and schemata

When we represent relationships in the semantic nets, as a schema, we get the new type of structured object which is named frame. A frame consists of a framename and

Table 6.1 An example of a frame

Framename	person
slot 1	isa: Student
slot 2	name: Ayşe
slot 3	related: Ahmet
slot 4	high school: Normal
slot 5	anayasa: BA

slots. A framename is name of initial node and slots are the named attributes. Slots are filled with values for particular instances.

As an example, we can re-represent the information about Ayse in the example used in section 3.1., in Table 6.1 as a frame.

We can incorporate subtype information in frames by using 'isa' slots. An 'isa' slot refers to another frame. In above example, 'isa' slot refers the 'student' frame. 'Student' frame stores information about students such as 'father name', 'address'.



CHAPTER SEVEN

SAMPLE PROGRAM

1. Introduction

In this chapter, we are going to give a sample program for data mining. For this program, we used a database about the students of a faculty.

The aim of this program is to investigate whether there is a relation in between the high school types, together with the family income level and their level of success in certain courses.

We chose three courses of freshman students, namely ANAYASA, HUKUK BAŞLANGICI and ROMA HUKUKU. These three courses names together with type of high school and income level of student's family constituted our five attributes. Student identification number is used to define the student as the sixth and unique attribute.

To discuss the result and for simplicity in explanations we limited the number of attributes, which in the helped us to visualize the result easily. Additionally, the program also searches whether the success in a certain course has a relation with the success in the other courses.

2. Database

In the database, we have attributes names as 'NO', 'GELIR', 'OKUL', 'HUKBAS', 'ANAYASA' and 'ROMA'. 'NO' is used for the student's identification numbers, 'GELIR' is used for the incomes of student's family, 'OKUL'

is used for the type of the high school, 'HUKBAS' is used for the grades of the course which is named 'HUKUK BAŞLANGICI', 'ANAYASA' is used for the grades of the course which is named 'ANAYASA HUKUKU' and 'ROMA' is used for the grades of the course which is named 'ROMA HUKUKU'.

This database is a relational database which has only one table. The domain of the attribute 'NO' contains identification numbers of students and all values are unique. 'NO' is only attribute which has unique values.

The domain of 'GELIR' has a large scale. But we chose some arbitrary intervals. These intervals are ' $GELIR \leq 150000000$ ', ' $150000000 < GELIR \leq 250000000$ ', ' $250000000 < GELIR \leq 500000000$ ' and ' $500000000 < GELIR$ '.

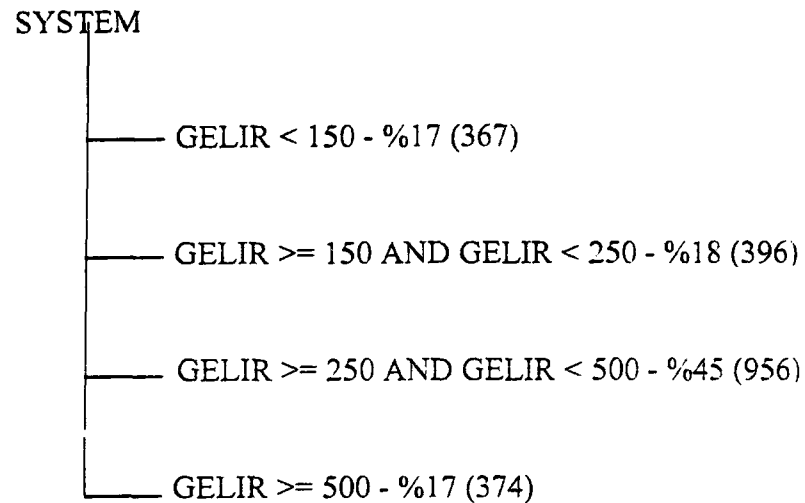
For the 'OKUL', we chose four types of high schools. These are 'NORMAL', 'ANADOLU', 'TEKNIK' and 'FEN'. The other attributes 'HUKBAS', 'ANAYASA' and 'ROMA' have only two different values, '1' for success and '0' for fail.

We needed to make an inquiry among the students for the values of the attribute 'GELIR', because, the faculty administration has not got this information.

3. Rules and classes

We used a decision tree for the representation of inferred knowledge. The root of the decision tree is always chosen as 'SYSTEM'. 'SYSTEM' represents all tuples (or all students) in the database.

'SYSTEM' can be classified according to any one of the attributes. That attribute can be chosen by the user. Data mining program can classify the students according to the chosen attribute.



**Figure 7.1 : The decision tree formed by the program
when attribute 'GELIR' is chosen**

In Figure 7.1, there are four classes. For the class 1. the rule is 'GELIR < 150' and there are 367 students in the class 1. For the class 2, the rule is 'GELIR >= 150 and GELIR < 250' and there are 396 students in the class 2. For the class 3. the rule is 'GELIR >= 250 and GELIR < 500' and there are 956 students in the class 3. For the class 4, the rule is 'GELIR >= 500' and there are 374 students in the class 2.

User can choose at most four attributes. The program is capable of adapting itself according to the wishes of the user, i.e. if user chooses 'GELIR' as the first attribute and then 'OKUL' as the second attribute or vice versa; the program will show the results accordingly.

In Figure 7.2, it can be seen that there are sixteen classes. We can define the rules of classes as " GELIR < 150 AND OKUL = 'NORMAL' " for the class 1, "GELIR<150 AND OKUL = 'ANADOLU' " for the class 2, "GELIR < 150 AND OKUL = 'TEKNIK' " for the class 3, and so on. The class 1 has 316 tuples (or students), the class 2 has 0 tuples and the class 3 has 51 tuples.

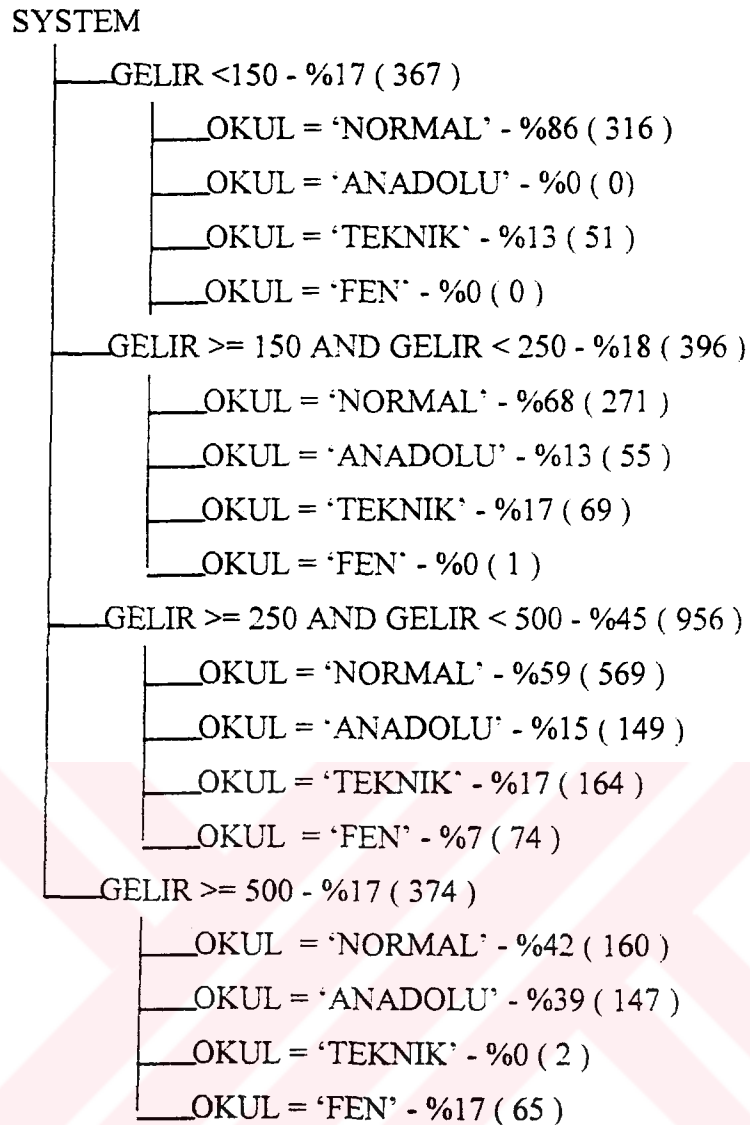


Figure 7.2. The decision tree formed by the program when attributes 'GELIR' and 'OKUL' are chosen

In the figure 7.3, we chose four attributes in order which are named 'GELIR', 'OKUL', 'HUKBAS' and 'ANAYASA'. 64 classes are created by the program. The rules of classes can be written as " GELIR < 150 AND OKUL = 'NORMAL' AND HUKBAS = 0 AND ANAYASA = 0 " for the class1, "GELIR < 150 AND OKUL = 'NORMAL' AND HUKBAS = 0 AND ANAYASA = 1 " for the class 2, and so on. The class 1 has 50 tuples and the class 2 has 168 tuples.

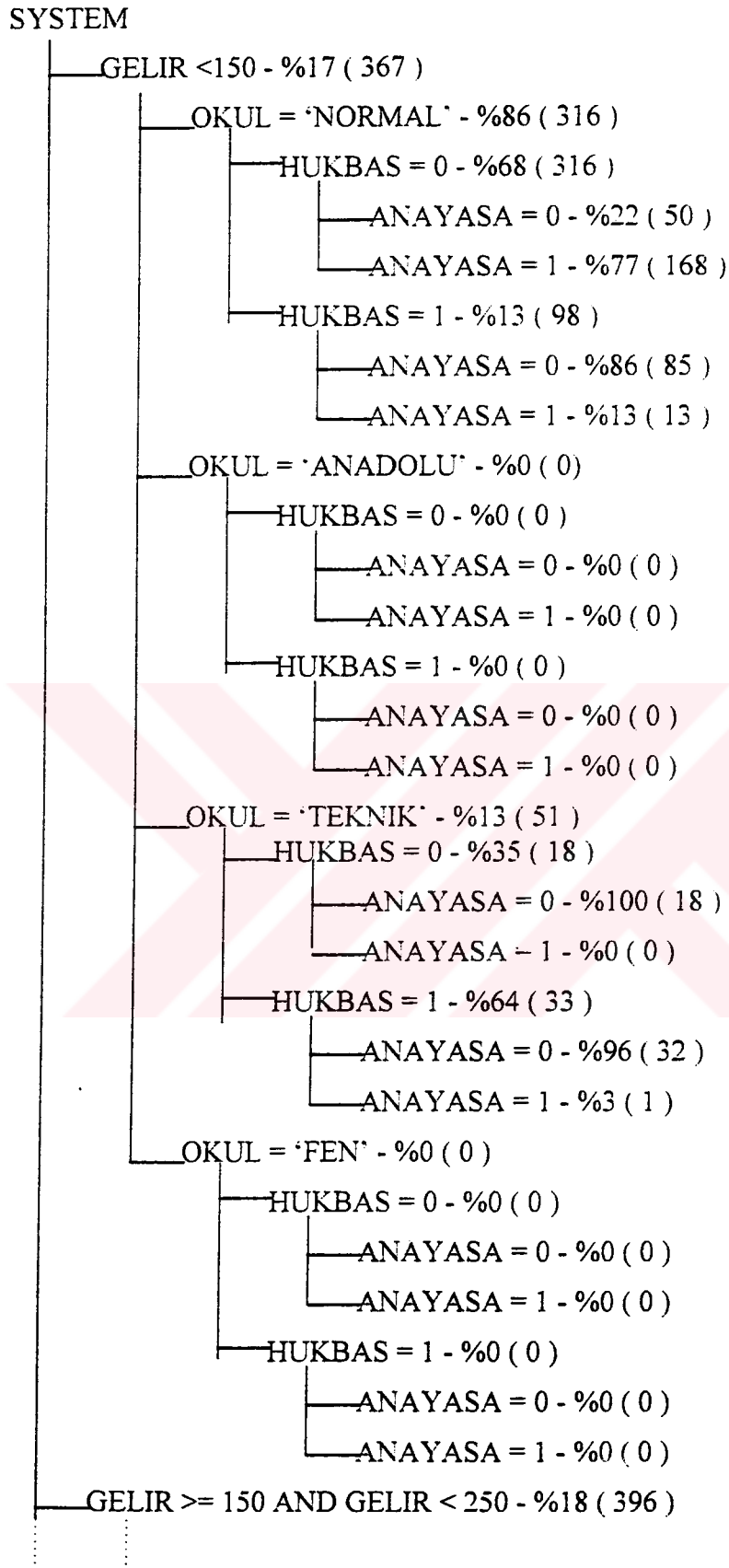


Figure 7.3. The decision tree formed by the program when attributes 'GELIR', 'OKUL', 'HUKBAS' and 'ANAYASA' are chosen

CONCLUSIONS

Despite its rapid growth, the KDD field is still in its infancy. There are many challenges to overcome, but some successes have been achieved. Because the potential payoffs of KDD applications are high, there has been a rush to offer products and services in the market. A great challenge facing the fields is how to avoid the kind of false expectations plaguing other nascent (and related) technologies (e.g., artificial intelligence and neural networks). It is the responsibility of researchers and practitioners in this field to ensure that the potential contributions of KDD are not overstated and that users understand the true nature of the contributions along with their limitations.

Fundamental problems at the heart of the field remain unsolved. For example, the basic problems of statistical inference and discovery remains as difficult and challenging as they always have been. Capturing the art of analysis and the ability of the human brain to synthesize new knowledge from data is still unsurpassed by any machine. However, the volumes of data to be analyzed make machines a necessity. This niche for using machines as an aid to analysis and the hope that the massive datasets contain nuggets of valuable knowledge drive interest in the field. Bringing together a set of varied fields, KDD creates fertile ground for the growth of new tools for managing, analyzing, and eventually gaining the upper hand over the flood of data facing modern society. The fact that the field is driven by strong social and economic needs is the impetus to its continued growth. The reality check of real applications will act as a filter to sift the good theories and techniques from those less useful.

REFERENCES

- Brachman, R.J., Khabaza, T., Kloesgen, W., Piatetsky-Shapiro, G. & Simoudis, E. (1996). Mining business databases. Communication of the Acm. 39, 42-48.
- Chen, M.S., Han, J. & Yu, P.S. (1996). Data mining: An overview from a database perspective. IEEE Transaction on Knowledge and Data Engineering. 8, 866-883.
- Fayyad, U., Haussler, D. & Stolorz, P. (1996). Mining scientific data. Communication of the Acm. 39, 51-57.
- Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. Communication of the Acm. 39, 27-34.
- Fayyad, U. & Uthurusamy, R. (1996). Data mining and knowledge discovery in databases. Communication of the Acm. 39, 24-26.
- Glymour, C., Madigan, D., Pregibon, D. & Smyth, P. (1996). Statistical inference and data mining. Communication of the Acm. 39, 35-41.
- Holsheimer, M. & Siebes, A. (1991). Data minig: The search for knowledge in databases. Amsterdam.
- Imielinski, T. & Mannila, H. (1996). A database perspective on knowledge discovery. Communication of the Acm. 39, 58-64.
- Inmon, W.H. (1996). The data warehouse and data mining. Communication of the Acm. 39, 49-50.
- Jain, A.K. & Dubes, R.C. (1988). Algorithms for clustering data. New Jersey: Prentice-Hall.
- Knorr, E.M. & Ng, R.T. (1996). Finding aggregate proximity relationships and commonalities in spatial data mining. IEEE Transaction on Knowledge and Data Engineering. 8, 884-897.

Özkarahan. E. (1997). Database Management : Concepts, Design, and Practise (2nd ed). İzmir: Saray Medical Publication.



T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ