

## Chapter 2

# Theoretical Preliminaries

As mentioned in Chap. 1, some fundamental concepts strongly relate to rule based systems and machine learning, including discrete mathematics, statistics, if-then rules, algorithms, logic and statistical measures of rule quality. This chapter illustrates these concepts in detail. In addition, this chapter also describes a unified framework for construction of single rule based classification systems, as well as the way to construct an ensemble rule based classification systems by means of a system of systems.

### 2.1 Discrete Mathematics

Discrete mathematics is a branch of mathematical theory, which includes three main topics, namely mathematical logic, set theory and graph theory. In this book, rule learning methods introduced in Chap. 3 are strongly based on Boolean logic, which is a theoretical application of mathematical logic in computer science. As mentioned in Sect. 1.1, a rule based system consists of a set of rules. In other words, rules are basically stored in a set, which is referred to as rule set. In addition, the data used in machine learning tasks is usually referred to as a dataset. Therefore, set theory is also strongly related to the materials in this book. The development of rule based networks, which is introduced in Chap. 5, is fundamentally based on graph theory. On the basis of above description, this subsection introduces in more detail the three topics as part of discrete mathematics with respects to their concepts and connections to the context of this book.

Mathematical logic includes the propositional connectives namely conjunction, disjunction, negation, implication and equivalence. Conjunction is also referred to as AND logic in computer science and denoted by  $F = a \wedge b$ . The conjunction could be illustrated by the truth table (Table 2.1).

Table 2.1 essentially implies that the output is positive if and only if all inputs are positive in AND logic. In other words, if any one of the inputs is negative, it would result in a negative output. In practice, the conjunction is widely used to make judgments especially on safety critical judgment. For example, it can be used for security check systems and the security status is positive if and only if all

**Table 2.1** Conjunction truth table

a	b	F
0	0	0
0	1	0
1	0	0
1	1	1

**Table 2.2** Disjunction truth table

a	b	F
0	0	0
0	1	1
1	0	1
1	1	1

parameters relating to the security are positive. In this book, the conjunction is typically used to judge if a rule is firing and more details about it are introduced in Sect. 1.4.3.

Disjunction is also referred to as OR logic in computer science and denoted by  $F = a \vee b$ . The disjunction is illustrated by the truth table (Table 2.2).

Table 2.2 essentially implies that the output would be negative if and only if all of the inputs are negative in OR logic. In other words, if any one of the inputs is positive, then it would result in a positive output. In practice, it is widely used to make judgments on alarm system. For example, an alarm system would be activated if any one of the parameters appears to be negative.

Implication is popularly used to make deductions, and is denoted by  $F = a \rightarrow b$ . The implication is illustrated by the truth table (Table 2.3).

Table 2.3 essentially implies that ‘a’ is defined as an antecedent and ‘b’ as a consequent. In this context, it supposes that the consequent would be deterministic if the antecedent is satisfied. In other words, ‘a’ is seen as the adequate but not necessary condition of ‘b’, which means if ‘a’ is true then ‘b’ will definitely be true, but b may be either true or false otherwise. In contrast, if ‘b’ is true, it is not necessarily due to that ‘a’ is true. This can also be proved as follows:

$$F = a \bullet b \Leftrightarrow \neg a \vee b$$

The notation  $\neg a \vee b$  is illustrated by the truth table (Table 2.4). In particular, it can be seen from the table that the output is negative if and only if ‘a’ provides a positive input but ‘b’ provides a negative one.

Table 2.4 essentially implies that the necessity condition that the output is negative is to have ‘a’ provide a positive input. This is because the output will definitely be positive when ‘a’ provides a negative input, which makes ‘ $\neg a$ ’ provide a positive input. In contrast, if ‘a’ provides a positive input and ‘b’ provides a negative input, then the output will be negative.

It can be seen from Tables 2.3 and 2.4 that the outputs from the two tables are exactly same. Therefore, Table 2.3 indicates that if an antecedent is satisfied then

**Table 2.3** Implication truth table

a	b	F
0	0	1
0	1	1
1	0	0
1	1	1

**Table 2.4** Negation truth table

a	b	$\neg a$	F
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

the consequent can be determined. Otherwise, the consequent would be non-deterministic. In this book, the concept of implication is typically used in the form of if-then rules for predicting classes. The concept of if-then rules is introduced in Sect. 2.3.

Besides, negation and equivalence are actually not applied to the research methodology in this book. Therefore, they are not introduced in detail here, but the interested reader can find these two concepts in [1].

Set theory is another part of discrete mathematics as mentioned earlier. A set is defined as a collection of elements. The elements maybe numbers, points and names etc., which are not ordered nor repetitive, i.e. the elements can be stored in any order and are distinct from each other. As introduced in [2, 3], an element ‘e’ has a membership in a set ‘S’, which is denoted by ‘ $e \in S$ ’ and it is said that element ‘e’ belongs to set ‘S’. The fact that the element ‘e’ is not a member of set ‘S’ is denoted by ‘ $e \notin S$ ’ and it is said that element ‘e’ does not belong to set ‘S’. In this book, set theory is used in the management of data and rules, which are referred to as data set and rule set respectively. A data set is used to store data and each element represents a data point. In this book, a data point is usually referred to as an instance. A rule set is used to store rules and each element represents a rule. In addition, a set can have a number of subsets depending on the number of elements. The maximum number of subsets for a set would be  $2^n$ , where  $n$  is the number of elements in the set. There are also some operations between sets such as union, intersection and difference, which are not relevant to the materials in this book. Therefore, the concepts relating to these operations are not introduced here—more details are available in [1, 3].

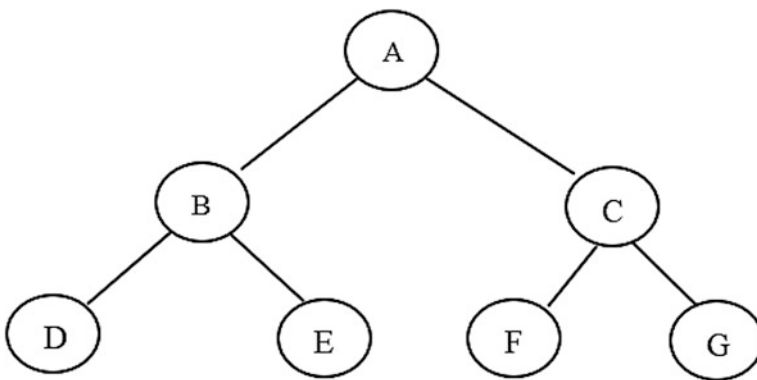
On the other hand, relations can be defined between sets. A binary relation exists when two sets are related. For example, there are two sets denoted as ‘Student’ and ‘Course’ respectively. In this context, there would be a mapping from students and courses, and each mapping is known as an ordered pair. For example, each student can register on one course only, but a course could have many students or no students, which means that each element in the set ‘Student’ is only mapped to one element in the set ‘Course’, but an element in the latter set may be mapped to many

elements in the former set. Therefore, this is a many-to-one relation. This type of relation is also known as a function. In contrast, if the university regulations allow that a student may register on more than one course, the relation would become many-to-many and is not a function any more. Therefore, a function is generally defined as a many-to-one relation. In the above example, the set ‘Student’ is regarded as the domain and the set ‘Course’ as range. In this book, each rule in a rule set actually acts as a particular function to reflect the mapping from input space (domain) to output space (range).

Graph theory is also a part of discrete mathematics as mentioned earlier in this subsection. It is popularly used in data structures such as binary search trees and directed or undirected graphs. A tree typically consists of a root node and some internal nodes as well as some leaf nodes as illustrated in Fig. 2.1. In this figure, node A is the root node of the tree; node B and C are two internal nodes; and node D, E, F and G are four leaf nodes. A tree could be seen as a top-down directed graph. This is because the search strategy applied to trees is in top-down approach from the root node to the leaf nodes. The search strategy could be divided into two categories: depth first search and breadth first search. In the former strategy, the search is going through in the following order:  $A \rightarrow B \rightarrow D \rightarrow E \rightarrow C \rightarrow F \rightarrow G$ . In contrast, in the latter strategy, the search would be in a different order:  $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G$ . In this book, the tree structure is applied to the concept of decision tree to graphically represent a set of if-then rules. More details about this are introduced in Chap. 5.

In contrast to trees, there is also a type of horizontally directed graphs in one/two way(s) as illustrated in Figs. 2.2 and 2.3. For example, a feed-forward neural network is seen as a one way directed graph and a feedback neural network as a two way directed graph.

In a directed graph, what could be judged is on the reachability between nodes depending on the existence of connections. For example, looking at Fig. 2.2, it can only be judged that it is reachable from node A to node C but unreachable in the



**Fig. 2.1** Example of tree structure

opposite way. This is because there is only a one way connection from node A to node C. In contrast, there is a two way connection between node A and node C through looking at Fig. 2.3. Therefore, it can be judged that it is reachable between the two nodes, i.e. it is reachable in both ways ( $A \rightarrow C$  and  $C \rightarrow A$ ). In this book, the concept of directed graphs is applied to a special type of rule representation known as rule based network for the purpose of predictive modelling. Related details are introduced in Chap. 5.

In addition, a graph could also be undirected, which means that in a graphical representation the connections between nodes would become undirected. This concept is also applied to network based rule representation but the difference to application of directed graphs is that the purpose is for knowledge representation. More details about this are introduced in Chap. 5.

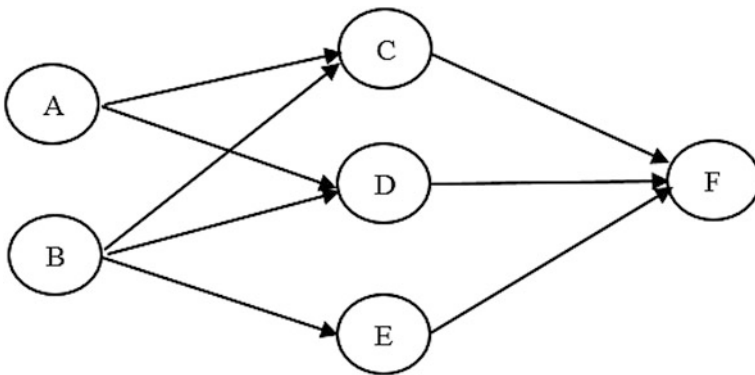


Fig. 2.2 Example of one way directed graph

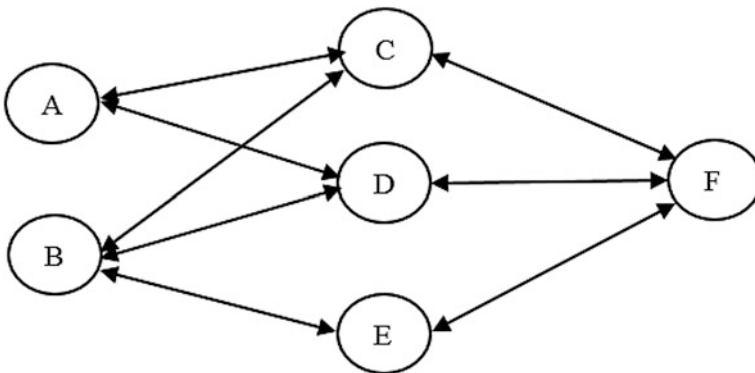


Fig. 2.3 Example of two way directed graph

## 2.2 Probability Theory

Probability theory is another branch of mathematics, which is a concept involved in all type of activities [4]. Probability is seen as a measure of uncertainty for a particular event. In general, there are two extreme cases. The first one is that if an event  $A$  is exact, then the probability of the event, denoted by  $P(A)$ , is equal to 1. The other case is that if the event is impossible, then the corresponding probability would be equal to 0. In reality, most events have a random behavior and their corresponding probabilities would be ranged between 0 and 1. These events typically include independent events and mutually exclusive events.

Independent events generally mean that for two or more events the occurrence of one does not affect that of the other(s). However, the events will be mutually exclusive if the occurrence of one event results in the non-occurrence of the other(s). In addition, there are also some events that are neither independent nor mutually exclusive. In other words, the occurrence of one event may result in the occurrence of the other(s) with a probability. The corresponding probability is referred to as conditional probability, which is denoted by  $P(A|B)$ . The  $P(A|B)$  is pronounced as ‘the probability of  $A$  given  $B$  as a condition’. According to Bayes theorem [5],  $P(A)$  is seen as a prior probability, which indicates the pre-degree of certainty for event  $A$ , and  $P(A|B)$  as a posterior probability, which indicates the post-degree of certainty for event  $A$  after taking into consideration event  $B$ . In this book, the concept of probability theory introduced above is related to the essence of the methods for rule generation introduced in Chap. 3. In addition, the concept is also related to an information theoretic measure called J-measure, which is discussed in Sect. 2.6.

Probability theory is typically jointly used with statistics. For example, it can well contribute to the theory of distribution [4] with respect to probability distribution. As mentioned in [4], a probability distribution is often transformed from frequency distribution. When different events have the same probability, the probability distribution is in the case of normal distribution. In the context of statistics, normal distribution occurs while all possible outcomes have the same frequency resulting from a sampling based investigation. Probability distributions also help predict the expected outcome out of all possible outcomes in a random event. This could be achieved by weighted majority voting, while the random event is discrete, or by weighted averaging, while the event is continuous. In the above context, probability is actually used as the weight and the expected outcome is referred to as mathematical expectation. In addition, the probability distribution also helps measure the approximate distance between expected outcome and actual outcome, while the distance among different outcomes is precise such as rating from 1 to 5. This could be achieved by calculating the variance or standard deviation to reflect the volatility with regard to the possible outcome. In this book, the probability distribution is related to a technique of information theory, which is known as entropy and used as a measure of uncertainty in classification. In addition, the concept on mathematical expectation is used to measure the expected accuracy

by random guess in classification and variance/standard deviation can be used to measure the randomness of an algorithm of ensemble learning.

## 2.3 If-then Rules

As mentioned in Sect. 1.1, rule based system typically consists of a set of if-then rules. Ross stated in [1] that there are many different ways for knowledge representation in the area of artificial intelligence but the most popular one would perhaps be in the form of if-then rules denoted by the expression: IF cause (antecedent) THEN effect (consequent).

The expression above typically indicates an inference that if a condition (cause, antecedent) is known then the outcome (effect, consequent) can be derived [1]. Gegov introduced in [6] that both the antecedent and the consequent of a rule could be made up of multiple terms (inputs/outputs). In this context, an antecedent with multiple inputs that are linked by ‘and’ connectives is called a conjunctive antecedent, whereas the inputs that are linked by ‘or’ connectives would make up a disjunctive antecedent. The same concept is also applied to rule consequent. In addition, it is also introduced in [6] that rules may be conjunctive, if all of the rules are connected by logical conjunction, or disjunctive, if the rules are connected by logical disjunction. On the other hand, a rule may be inconsistent, which indicates that the antecedent of a rule may be mapped to different consequents. In this case, the rule could be expressed with a conjunctive antecedent and a disjunctive consequent.

In this book, if-then rules are used to make prediction in classification tasks. In this context, each of the rules is referred to as a classification rule, which can have multiple inputs, but only a single output. In a classification rule, the consequent with a single output represents the class predicted and the antecedent with a single/multiple input(s) represents the adequate condition to have this class predicted. A rule set that is used to predict classes consists of disjunctive rules which may be overlapped. This means that different rules may have the same instances covered. However, if the overlapped rules have different consequents (classification), it would raise a problem referred to as conflict of classification. In this case, conflict resolution is required to solve the problem according to some criteria such as weighted voting or fuzzy inference [1]. When a rule is inconsistent, it would result in uncertainty in classification. This is because the prediction of class becomes non-deterministic when this problem arises. More details about conflict resolution and dealing with inconsistent rules are introduced in Chap. 3.

Another concept relating to if-then rules is known as a rule base. In general, a rule base consists of a number of rules which have common input and output variables. For example, a rule base has two inputs:  $x_1$  and  $x_2$  and one output  $y$  as illustrated by Fig. 2.4.

If  $x_1$ ,  $x_2$  and  $y$  all belong to  $\{0, 1\}$ , the rule base can have up to four rules as listed below:



**Fig. 2.4** Rule base with inputs  $x_1$  and  $x_2$  and output  $y$

If  $x_1 = 0$  and  $x_2 = 0$  then  $y \in \{0, 1\}$

If  $x_1 = 0$  and  $x_2 = 1$  then  $y \in \{0, 1\}$

If  $x_1 = 1$  and  $x_2 = 0$  then  $y \in \{0, 1\}$

If  $x_1 = 1$  and  $x_2 = 1$  then  $y \in \{0, 1\}$

In practice, rule bases can be used to effectively and efficiently manage rules with respects to their storage and retrieval. For example, if a particular rule is searched for, it could be efficiently retrieved by locating at the rule base in which the rule is found. This is a significant difference to rule set for retrieval purpose. As mentioned earlier in this section, set is used to store a collection of elements which are not ordered nor grouped properly. From this point of view, it is not efficient to look for a particular rule in a rule set. The only way to deal with that is to linearly go through the rules one by one in the rule set until the target rule is found. In the worst case, it may be required to go through the whole set due to that the target rule is restored as the last element of the rule set. Therefore, the use of rule base would improve the efficiency in predicting classes on unseen instances in testing stage. More details about the use of rule bases are introduced in Chap. 8.

## 2.4 Algorithms

Aho et al. defined in [3] that “algorithm is a finite sequence of instructions, each of which has a clear meaning and can be performed with a finite amount of effort in a finite length of time”. In general, an algorithm acts as a step by step procedure for problem solving. An algorithm may have no inputs but must have at least one output with regard to solving a particular problem. In practice, a problem can usually be solved by more than one algorithm. In this sense, it is necessary to make comparison between algorithms to find the one which is more suitable to a particular problem domain. An algorithm could be evaluated against the following aspects:

Accuracy, which refers to the correctness in terms of correlation between inputs and outputs.

Efficiency, which refers to the computational cost required.

Robustness, which refers to the tolerance to incorrect inputs.

Readability, which refers to the interpretability to people.



Accuracy would usually be the most important factor in determining whether an algorithm is chosen to solve a particular problem. It can be measured by providing the inputs and checking the outputs.

Efficiency is another important factor to measure if the algorithm is feasible in practice. This is because if an algorithm is computationally expensive then the implementation of the algorithm may be crashed on a hardware device. Efficiency of an algorithm can usually be measured by checking the time complexity of the algorithm in theoretical analysis. In practice, it is usually measured by checking the actual runtime on a machine.

Robustness can usually be measured by providing a number of incorrect inputs and checking to what extent the accuracy with regard to outputs is affected.

Readability is also important especially when an algorithm is theoretically analyzed by experts or read by practitioners for application purpose. This problem can usually be solved by choosing a suitable representation for the algorithm to make it easier to read. Some existing representations include flow chart, UML activity diagram, pseudo code, text and programming language.

This book addresses these four aspects in Chaps. 3, 4, 5, 6 and 7 in the way of theoretical analysis, as well as algorithm representation with regard to algorithm analysis.

## 2.5 Logic

Ross stated in [1] that logic is a small part of the capability of human reasoning, which is used to assist people in making decisions or judgments. Section 2.1 introduced mathematical logic which is also referred to as Boolean logic in computer science. As mentioned in Sect. 2.1, in the context of Boolean logic, each variable is only assigned a binary truth value: 0 (false) or 1 (true). It indicates that reasoning and judgment are made under certainty resulting in deterministic outcomes. From this point of view, this type of logic is also referred to as deterministic logic. However, in reality, people usually can only make decisions, and apply judgment and reasoning under uncertainty. Therefore, the other two types of logic, namely probabilistic logic and fuzzy logic, are used more popularly, both of which can be seen as an extension of deterministic logic. The main difference is that the truth value is not binary but continuous between 0 and 1. The truth value implies a probability of truth between true and false in probabilistic logic and a degree of that in fuzzy logic. The rest of the subsection introduces the essence of the three types of logic and the difference between them as well as how they are linked to the concept of rule based systems.

Deterministic logic deals with any events under certainty. For example, when applying deterministic logic for the outcome of an exam, it could be thought that a student will exactly pass or fail a unit. In this context, it means the event is certain to happen.

Probabilistic logic deals with any events under probabilistic uncertainty. For the same example about exams, it could be thought that a student has an 80 % chance to pass, i.e. 20 % chance to fail, for a unit. In this context, it means the event is highly probable to happen.

Fuzzy logic deals with any events under non-probabilistic uncertainty. For the same example about exams, it could be thought that a student has 80 % factors of passing, i.e. 20 % factors of failing, for a unit with regard to all factors in relation to the exam. In this context, it means the event is highly likely to happen.

A scenario is used to illustrate the above description as follows: students need to attempt the questions on four topics in a Math test. They can pass if and only if they pass all of the four topics. For each of the topics, they have to get all answers correct to pass. The exam questions do not cover all aspects that students are taught, but should not be outside the domain nor be known to students. Table 2.5 reflects the depth of understanding of a student in each of the topics.

In this scenario, deterministic logic is not applicable because it is never deterministic with regard to the outcome of the test. In other words, deterministic logic is not applicable in this situation to infer the outcome (pass/fail).

In probabilistic logic, the depth of understanding is supposed to be the probability of the student passing. This is because of the assumption that the student would exactly gain full marks for which questions the student is able to work out. Therefore, the probability of passing would be:  $p = 0.8 \times 0.6 \times 0.7 \times 0.2 = 0.0672$ .

In fuzzy logic, the depth of understanding is supposed to be the weight of the factors for passing. For example, for topic 1, the student has 80 % factors for passing but it does not imply that the student would have 80 % chance to pass. This is because in reality the student may feel unwell mentally, physically and psychologically. All of these issues may make it possible that the student will make mistakes as a result of that the student may fail to gain marks for which questions that normally he/she would be able to work out. The fuzzy truth value of passing is  $0.2 = \min (0.8, 0.6, 0.7, 0.2)$ . In this context, the most likely outcome for failing would be that the student only fails one topic resulting in a failure of Math. The topic 4 would be obviously the one which is most likely to fail with the fuzzy truth value 0.8. In all other cases, the fuzzy truth value would be less than 0.8. Therefore, the fuzzy truth value for passing is  $0.2 = 1 - 0.8$ .

In the context of set theory, deterministic logic implies that a crisp set that has all its elements fully belong to it. In other word, each element has a full membership to the set. Probabilistic logic implies that an element may be randomly allocated to one of a finite number of sets with normal distribution of probability. Once the element has been allocated to a particular set, then it has a full membership to the set. In other words, the element is eventually allocated to one set only. Fuzzy logic implies that a set is referred to as fuzzy set because each element may not have a full

**Table 2.5** Depth of understanding for each topic

Topic 1	Topic 2	Topic 3	Topic 4
80 (%)	60 (%)	70 (%)	20 (%)

membership to the set. In other words, the element belongs to the fuzzy set to a certain degree.

In the context of rule base systems, a deterministic rule based system would have a rule either fire or not. If it fires, the consequence would be deterministic. A probabilistic rule based system would have a firing probability for a rule. The consequence would be probabilistic depending on posterior probability of it given specific antecedents. A fuzzy rule based system would have a firing strength for a rule. The consequence would be weighted depending on the fuzzy truth value of the most likely outcome. In addition, fuzzy rule based systems deal with continuous attributes by mapping the values to a number of linguistic terms according to the fuzzy membership functions defined. More details about the concepts on rule based systems outlined above are introduced in Chap. 5.

## 2.6 Statistical Measures

In this book, some statistical measures are used as heuristics for development of rule learning algorithms and evaluation of rule quality. This subsection introduces some of these measures, namely entropy, J-measure, confidence, lift and leverage.

Entropy is introduced by Shannon in [7], which is an information theoretic measure of uncertainty. Entropy  $E$  can be calculated as illustrated in Eq. (2.1):

$$E = - \sum_{i=0}^n p_i \cdot \log_2 p_i \quad (2.1)$$

where  $p$  is read as probability that an event occurs and  $i$  is the index of the corresponding event.

J-measure is introduced by Smyth and Goodman in [8], which is an information theoretic measure of average information content of a single rule. J-measure is essentially the product of two terms as illustrated in Eq. (2.2):

$$J(Y, X = x) = P(x) \cdot j(Y, X = x) \quad (2.2)$$

where the first term  $P(x)$  is read as the probability that the rule antecedent (left hand side) occurs and considered as a measure of simplicity [8]. In addition, the second term is read as j-measure, which is first introduced in [9] but later modified in [8] and considered as a measure of goodness of fit of a single rule [8]. The j-measure is calculated as illustrated in Eq. (2.3):

$$j(Y, X = x) = P(y|x) \cdot \left( \frac{P(y|x)}{P(y)} \right) + \left( 1 - P(y|x) \cdot \left( \frac{1 - P(y|x)}{1 - P(y)} \right) \right) \quad (2.3)$$

where  $P(y)$  is read as prior probability that the rule consequent (right hand side) occurs and  $P(y|x)$  is read as posterior probability that the rule consequent occurs given the rule antecedent as the condition.

In addition, j-measure has an upper bound referred to as jmax as indicated in [8] and illustrated in Eq. (2.4):

$$j(Y, X = x) \leq \max(P(y|x) \cdot \left(\frac{1}{P(y)}\right), \left(1 - P(y|x) \cdot \left(\frac{1}{1 - P(y)}\right)\right)) \quad (2.4)$$

However, if it is unknown to which class the rule is assigned as its consequent, then the j-measure needs to be calculated by taking into account all possible classes as illustrated in Eq. (2.5):

$$j(Y, X = x) = \sum_{i=0}^n P(y_i|x) \cdot \left(\frac{P(y_i|x)}{P(y_i)}\right) \quad (2.5)$$

In this case, the corresponding jmax is calculated in the way illustrated in Eq. (2.6):

$$j(Y, X = x) \leq \max_i \left( P(y_i|x) \cdot \left(\frac{1}{P(y_i)}\right) \right) \quad (2.6)$$

Confidence is introduced in [10], which is considered as predictive accuracy of a single rule, i.e. to what extent the rule consequent is accurate while the rule antecedent is met. The confidence is calculated as illustrated in Eq. (2.7):

$$Conf = \frac{P(x, y)}{P(x)} \quad (2.7)$$

where  $P(x, y)$  is read as the joint probability that the antecedent and consequent of a rule both occur and  $P(x)$  is read as prior probability as same as used in J-measure above.

Lift is introduced in [11], which measures to what extent the actual frequency of joint occurrence for the two events X and Y is higher than expected if X and Y are statistically independent [12]. The lift is calculated as illustrated in Eq. (2.8):

$$Lift = \frac{P(x, y)}{P(x) \cdot P(y)} \quad (2.8)$$

where  $P(x, y)$  is read as the joint probability of x and y as same as mentioned above and  $P(x)$  and  $P(y)$  are read as the coverage of rule antecedent and consequent respectively.

Leverage is introduced in [13], which measures the difference between the actual joint probability of x and y and the expected one [12]. The leverage is calculated as illustrated in Eq. (2.9):

$$\text{Leverage} = P(x, y) - P(x) \cdot P(y) \quad (2.9)$$

where  $P(x, y)$ ,  $P(x)$  and  $P(y)$  are read as same as in Eq. (2.9) above.

A more detailed overview of these statistical measures can be found in [14, 15]. In this book, entropy is used as a heuristic for rule generation and J-measure is used for both rule simplification and evaluation. In addition, confidence, lift and leverage are all used for evaluation of rule quality. More details on this will be given in Chaps. 3, 4, 5 and 6.

## 2.7 Single Rule Based Classification Systems

As mentioned in Chap. 1, single rule based systems mean that a particular rule based system consists of one rule set only. If such rule based systems are used for classification, they can be referred to as single rule based classification systems.

In machine learning context, single rule based classification systems can be constructed by using standard rule learning algorithms such as ID3, C4.5 and C5.0. This kind of systems can also be constructed using ensemble rule based learning approaches.

In both ways mentioned above, rule based classification systems can be constructed by adopting a unified framework that was recently developed in [16]. This framework consists of rule generation, rule simplification and rule representation.

Rule generation means to generate a set of rules by using one or more rule learning algorithm(s). The generated rule set is eventually used as a rule based system for prediction making. However, as mentioned in Chap. 1, most of rule learning algorithms suffer from overfitting of training data, which means the generated rule set may perform a high level of accuracy on training instances but a low level of accuracy on testing instances. In this case, the set of rules generated need to be simplified by means of rule simplification and using pruning algorithms, which generally tends to reduce the consistency but improve the accuracy of each single rule. As introduced in Chap. 1, rule generation algorithms can be divided into two categories: divide and conquer and separate and conquer. In addition, pruning algorithms can be subdivided into two categories: pre-pruning and post-pruning. In this context, if the divide and conquer approach is adopted, rule simplification is taken to stop the growth of a branch in a decision tree if pre-pruning is adopted for the simplification of rules. Otherwise, the rule simplification would be taken to simplify each branch after a complete decision tree has been generated. On the other hand, if the separate and conquer approach is adopted, rule simplification is taken to stop the specialization of a single rule if pre-pruning is adopted for the simplification. Otherwise, the rule simplification would be taken to post-prune each single rule after the completion of its generation.

Rule representation means to represent a set of rules in a particular structure such as decision trees, linear lists and rule based networks. In general, appropriate

representation of rules enables the improvement of both interpretability and computational efficiency.

In terms of interpretability, a rule based system is required to make knowledge extracted from the system easier for people to read and understand. In other words, it facilitates the communication of the knowledge extracted from the system. On the other hand, the rules should allow to interpret knowledge in a great depth, in an explicit way. For example, when a system provides an output based on a given input, the reason why this output is derived should be explained in a straightforward way.

In terms of computational efficiency, a rule based system is required to make a quick decision in practice due to time critical aspects. In particular, rule representation is just like data structures which are used to manage data in different ways. In software engineering, different data structures usually lead to different levels of computational efficiency in some operations relating to data management such as insertion, update, deletion and search. As mentioned in Chap. 1, it is required to find the first firing rule as quickly as possible in order to make a quick prediction. Therefore, this could be seen as a search problem. As mentioned above, different data structures may provide different levels of search efficiency. For example, a collection of items stored in a linear list can only be searched linearly if these items are not given indexes. However, if the same collection of items is stored in a tree, then it is achievable to have a divide and conquer search. The former way of search would be in linear time whereas the latter way in logarithmic time. In this sense, efficiency in search of firing rules would also be affected by the structure of the rule set. It is also defined in [17] that one of the biases for rule based systems is ‘search bias’, which refers to the strategy used for the hypothesis search. In general, what is expected is to make it unnecessary to examine a whole rule set, but as few rule terms as possible.

Overall, the unified framework for construction of single rule based classification systems consists of three operations namely, rule generation, rule simplification and rule representation. In practice, the first two operations may be executed in parallel or sequentially depending on the approaches adopted. Rule representation is usually executed after the finalization of a rule set used as a rule based system. More details about these operations are presented in Chaps. 3, 4 and 5.

## 2.8 Ensemble Rule Based Classification Systems

As mentioned in Chap. 1, ensemble rule based classification systems mean that a particular rule based system consists of multiple rule sets, each of which could be seen as a single rule based systems. This defines a novel way of understanding ensemble rule based systems in the context of system theory [18]. In particular, each of the single rule based systems could be seen as a subsystem of the ensemble rule based system by means of a system of systems.

Ensemble rule based classification systems could be constructed by using ensemble learning approaches. Each of the single rule based classification systems can be constructed still based on the unified framework introduced in Sect. 2.1. In particular, for rule learning algorithms, ensemble learning can be adopted in the way that a base algorithm is used to learn from a number of samples, each of which results from the original training data through random sampling with replacement. In this case, there are  $n$  rule sets generated while  $n$  is the number of samples. In the testing stage, each of the  $n$  rule sets make an independent prediction on an unseen instance and their predictions are then combined to make the final prediction. The  $n$  rule sets mentioned above make up an ensemble rule based system for prediction purpose as mentioned in the literature [19]. A typical example of such systems is Random Forest which consists of a number of decision trees [20] and is usually helpful for decision tree learning algorithms to generate more accurate rule sets [21]. On the other hand, in order to construct ensemble rule based classification systems, ensemble learning can also be adopted in another way that multiple rule learning algorithms work together so that each of the algorithms generates a single rule set on the same training data. These generated rule sets, each of which is used as a single rule based classification systems, are combined to make up an ensemble rule based classification system.

The two ways to construct ensemble rule based systems mentioned above follow parallel learning approaches as introduced in Chap. 1. Ensemble rule based systems can also be constructed following sequential learning approaches. In particular, the same rule learning algorithm is applied to different versions of training data on an iterative basis. In other words, at each iteration, the chosen algorithm is used to generate a single rule set on the training data. The rule set is then evaluated on its quality by using validation data and each of the training instances is weighted to a certain degree based on its contribution to generating the rule set. The updated version of the training data is used at the next iteration. As the end, there is a number of rule sets generated, each of which is used as a single rule based classification system. These single systems make up the ensemble rule based classification systems.

Ensemble rule based systems can be advanced via different computing environments such as parallel, distributed and mobile computing environments.

Parallel computing can significantly improve the computational efficiency for ensemble learning tasks. In particular, as mentioned above, parallel learning can be achieved through the way that a number of samples are drawn on the basis of the training data, each of the sample is used to build a single rule based system by using the same rule learning algorithm. In this context, each of the drawn samples can be loaded into a core of a parallel computer and the same rule learning algorithm is used to build a single rule based systems on each core on the basis of the training sample loaded into the core. This is a popular approach known as parallel data mining [22, 23].

Distributed computing can make ensemble rule based systems more powerful in practical applications. For example, for some large companies or organizations, the web architecture is usually developed in the form of distributed database systems,

which means that data relating to the information of the companies or organizations is stored into their databases distributed in different sites. In this context, distributed data mining is highly required to process the data. In addition, ensemble rule based systems also motivate collaborations between companies through collaborations involved in ensemble learning tasks. In particular, the companies that collaborate with each other can share access to their databases. Each of the databases can provide a training sample for the employed rule learning algorithm to build a single rule based system. Each of the companies can have the newly unseen instances predicted using the ensemble rule based system that consists of a number of single rule based systems, each of which is from a particular database for one of the companies. Some popular distributed data mining techniques can be found in [24, 25]. Similar benefits also apply to mobile data mining such as Pocket Data Mining [26, 27].

More details about these ensemble learning approaches for construction of rule based systems is given in Chap. 6 in more depth.

## References

1. Ross, T.J.: Fuzzy logic with engineering applications, 2nd edn. Wiley, West Sussex (2004)
2. Schneider, S.: The B-Method: an introduction. Palgrave Macmillian, Basingstoke, New York (2001)
3. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: Data structures and algorithms. Addison-Wesley, Boston (1983)
4. Murdoch, J., Barnes, J.A.: Statistics: problems and solutions. The Macmillan Press Ltd, London and Basingstoke (1973)
5. Hazewinkel, M. (ed.): Bayes formula. Springer, Berlin (2001)
6. Gegov, A.: Advanced computation models for rule based networks, Portsmouth (2013)
7. Shannon, C.: A mathematical theory of communication. Bell Syst. Tech. J. **27**(3), 379–423 (1948)
8. Smyth, P., Rodney, G.M.: An information theoretic approach to rule induction from databases. IEEE Trans. Knowl. Data Eng. **4**(4), 301–316 (1992)
9. Blachman, N.M.: The amount of information that y gives about X. IEEE Transactions. Inf. Theory. **14**(1), 27–31 (1968)
10. Agrawal, R., Imilielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of ACM SIGMOD International Conference on Management of Data, Washington D.C (1993)
11. Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Tucson, Arizona (1997)
12. Hahsler, M.: A probabilistic comparison of commonly used interest measures for association rules. Available: [http://michael.hahsler.net/research/association\\_rules/measures.html](http://michael.hahsler.net/research/association_rules/measures.html) (2015) (Online)
13. Piatetsky-Shapiro, G.: Discovery, analysis, and presentation of strong rules. In: Piatetsky-Shapiro, G., Frawley, W.J. (eds.) Knowledge Discovery in Databases. MA, AAAI/MIT Press, Cambridge (1991)
14. Tan, P.-N., Kumar, V., Srivastava, J.: Selecting the right objective measure for association analysis. Inf. Syst. **29**(4), 293–313 (2004)



15. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: a survey. *ACM Computing Surveys*, vol. 38, no. 3 (2006)
16. Liu, H., Gegov, A., Stahl, F.: Unified framework for construction of rule based classification systems. In: Pedrycz, W., Chen, S. (eds.) *Information Granularity, Big Data and Computational Intelligence*, vol. 8, pp. 209–230. Springer, Berlin (2015)
17. Furnkranz, J.: Separate-and-conquer rule learning. *Artif. Intell. Rev.* **13**, 3–54 (1999)
18. Stichweh, R.: Systems theory. In: Badie, B.E.A. (ed.) *International Encyclopaedia of Political Science*, New York, Sage (2011)
19. Liu, H., Gegov, A.: Collaborative decision making by ensemble rule based classification systems. In: Pedrycz, W., Chen, S. (eds.) *Granular Computing and Decision-Making: Interactive and Iterative Approaches*, vol. 10, pp. 245–264. Springer, Berlin (2015)
20. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
21. Kononenko, I., Kukar, M.: *Machine learning and data mining: introduction to principles and algorithms*. Horwood Publishing Limited, Chichester, West Sussex (2007)
22. Li, J., Liu, Y., Liao, W.-K., Choudhary, A.: *Parallel data mining algorithms for association rules and clustering*. CRC Press, Boca Raton (2006)
23. Parthasarathy, S., Zaki, M.J., Ogihara, M., Li, W.: Parallel data mining for association rules on shared-memory systems. *Knowl. Inf. Syst.* **3**, 1–29 (2001)
24. Giannella, C., Bhargava, R., Kargupta, H.: Multi-agent systems and distributed data mining. In: *Cooperative Information Agents VIII*, Berlin (2004)
25. Datta, S., Bhaduri, K., Giannella, C., Wolff, R., Kargupta, H.: Distributed data mining in peer-to-peer networks. *IEEE. Internet. Comput.* **10**(4), 18–26 (2006)
26. Gaber, M.M., Stahl, F., Gomes, J.B.: *Pocket data mining: big data on small devices*, vol. 2. Springer, Switzerland (2014)
27. Gaber, M.: Pocket data mining: the next generation in predictive analytics. In: *Predictive Analytics Innovation Summit*, London (2012)

Rule Based Systems for Big Data

A Machine Learning Approach

Liu, H.; Gegov, A.; Cocea, M.

2016, XIII, 121 p. 38 illus., 5 illus. in color., Hardcover

ISBN: 978-3-319-23695-7