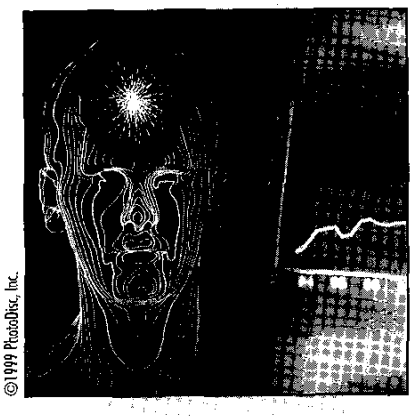


Discovering Knowledge from Medical Databases Using Evolutionary Algorithms

Learning Rules and Causal Structures for Capturing Patterns and Causality Relationships

Man Leung Wong[1], Wai Lam[2], Kwong Sak Leung[3], Po Shun Ngan[3], Jack C.Y. Cheng[4]

[1]Department of Information Systems, Lingnan University, Hong Kong
[2]Department of Systems Engineering and Engineering Management
[3]Department of Computer Science and Engineering
[4]Department of Orthopaedics and Traumatology
The Chinese University of Hong Kong

The increasing use of computers results in an explosion of information, making data mining an important research topic. Data can be best used if the knowledge hidden can be uncovered. Thus, there is a need for a way to automatically discover knowledge from data. Research in this area can be useful for many real-world problems. With computerization in hospitals, a huge amount of data has been collected. It is beneficial if these data can be analyzed automatically.

Data mining, sometimes referred to as knowledge discovery in databases (KDD), can be defined as the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [3]. KDD is an interactive and iterative process comprised of several steps. Data mining can be considered as one of the steps in the KDD process. It is the core of the KDD process, and thus the two terms are often used interchangeably.

The whole process of KDD consists of five steps. First, a selection is made to extract a relevant or a target data set from the database. Then, preprocessing is performed to remove noise and to handle missing data fields. Transformation is performed to reduce the number of variables under consideration. A suitable data mining algorithm is employed on the prepared data. Finally the result of the data mining is interpreted and evaluated. If the discovered knowledge is not satisfactory, these five steps will be iterated. The discovered knowledge is then applied in decision making.

In this article, we will introduce our approaches for discovering knowledge from two specific medical databases. Two different representations of knowledge, namely rules and causal structures, are learned. Rules capture interesting patterns and regularities in the database. Causal structures represented by Bayesian networks capture the causality relationships among the attributes. We employ evolutionary algorithms for these discovery tasks.

Evolutionary Algorithm Primer

Evolutionary algorithms simulate natural evolution to perform function optimization and machine learning. A potential solution to the problem is encoded as an *individual*. An evolutionary algorithm maintains a group of individuals, called the *population*, to explore the search space. A *fitness function* evaluates the performance of each individual to measure how close it is to the solution. The search space is explored by evolving new individuals. The algorithm is based on the Darwinian principle of evolution through natural selection; the fitter individual has a higher chance of survival and tends to pass on its favorable traits to its offspring. A "good" parent is assumed to be able to produce "good" or even better offspring. Thus, individuals with higher fitness scores have higher chances of producing offspring. New individuals are generated by applying *operators* that alter the underlying structure of these existing individuals. The process is repeated until the solution is found or the maximum number of iterations is reached. Evolutionary algorithms include genetic algorithms (GAs) [7, 9], genetic programming [10, 11], evolutionary programming [4, 5], and evolution strategy [17, 19].

Advanced evolutionary algorithms are used for knowledge discovery tasks. In particular, generic genetic programming

0739-5175/00/$10.00©2000IEEE

is employed as a rule-learning algorithm. Our approach for discovering causality relationships is based on evolutionary programming, which learns Bayesian network structures. To handle continuous attributes, we employ a GA to find a good discretization policy.

The Learning Tasks

Medical Databases

Our learning targets are two medical databases from the Orthopaedic Department of the Prince of Wales Hospital of Hong Kong. The first, the fracture database, consists of records of children with limb fractures admitted to the hospital in the period 1984-1996. These data can provide information for the analysis of child fracture patterns. This database has 6500 records and eight attributes, which are listed in Table 1.

Table 1. Attributes in the Fracture Database

Name	Type	Explanation
Sex	Nominal	Sex
Age	Numeric	Age (between 0 to 16 years old)
Admday	Date	Admission date (between year 1984 to 1996)
Stay	Numeric	Length of staying in hospital (in days)
Diagnosis	Nominal	Diagnosis of fracture based on the fracture location
Operation	Nominal	Operation
Surgeon	Nominal	Surgeon (null if no operation)
Side	Nominal	Side of fracture ("Left," "Right," "Both," or "Missing")

Table 2. Attributes in the Scoliosis Database

Name	Explanation
Sex	Sex
Age	Age
Lax	Joint Laxity (integer between 0 and 3)
1stCurveT1	Whether 1st curve started at vertebra T1
1stMCGreater	Whether the degree of 1st Major Curve is greater than the 2nd Major Curve
L4Tilt	Whether vertebra L4 is tilted
1stMCDeg	Degree of 1st Major Curve
2ndtMCDeg	Degree of 2nd Major Curve
1stMCApex	Apex of 1st Major Curve
2ndMCApex	Apex of 2nd Major Curve
Deg1	Degree of 1st Curve
Deg2	Degree of 2nd Curve
Deg3	Degree of 3rd Curve
Deg4	Degree of 4th Curve
Class	Scoliosis Classification (K-I, K-II, K-III, K-IV, K-V, TL or L)
Mens	Period of Menstruation
TSI	Trunk Shift (in cm), which measures the displacement of the curve
TSIDir	Trunk Shift Direction (null, left or right)
RI	Risser Sign (integer between 0 and 5), which measures the maturity of the patient
Treatment	Treatment (observation, surgery, or bracing)

The second database contains clinical records of scoliosis patients. A scoliosis patient has one or several curves in the spine. Curves with severe deformations are identified as major curves. The database stores measurements on the patients, such as the number of curves, the curve locations, degrees, and directions. It also records the age of the patient, the class of scoliosis, and the treatment. The database has about 500 records, with the attributes shown in Table 2.

Rule Learning

We investigate the task of discovering rules from these two databases. We make use of a rule representation that is easily understandable. A rule is a sentence of the form "if *antecedents*, then *consequent*." The antecedents specify certain characteristics of attributes. In general, the antecedent part is a conjunction of descriptions about attributes, while the consequent is a descriptor for a single attribute. Rule learning is the process of inducing rules from a set of training examples.

The accuracy or the confidence of a rule is the probability that the consequent occurs under the condition that the antecedents occur. If the accuracy is 100%, the rule is an exact rule. If the accuracy is near 100%, the rule is a strong rule. If the accuracy is not high but is already much larger than the average probability of the consequent, then the rule is a weak rule. A data mining approach should not discover only exact or strong rules, because weak rules may also provide useful knowledge.

Bayesian Network Learning

A Bayesian network [2, 8] is a different model used to represent probabilistic knowledge of data. It is a formal knowledge representation supported by the well-developed Bayesian probability theory. It captures the conditional probabilities between variables (i.e., attributes in the database) and focuses on causality relationships among variables. In many real-life situations, the data cannot be described completely by a few rules. Building a complete model for such a database is difficult and usually results in a complicated model. A Bayesian network can be a

complement to rules and, due to its graphical representation, is easily understandable. It has a well-developed mathematical model and can be used to perform reasoning under uncertainty.

Formally, a Bayesian network is a directed acyclic graph (DAG). Each node represents an attribute, and each edge represents a dependency between two nodes. An edge from node A to node B can represent a causality conveying the fact that the value of B depends on the value of A. The value of each variable is discrete. Each node is associated with a set of parameters. Let N_i denote a node and Π_{N_i} denote the set of parents of N_i. The parameters of N_i are conditional probability distributions in the form of $P(N_i|\Pi_{N_i})$, with one distribution for each possible instance of Π_{N_i}.

The main task of learning in a Bayesian network is to automatically find directed edges between the nodes, such that the network can best describe the causalities. Once the network structure is constructed, the conditional probabilities are calculated based on the data.

Rule Learning Using Generic Genetic Programming

We employ an advanced evolutionary algorithm called generic genetic programming (GGP) to discover rules from a database. GGP [20, 22, 23] is an extension of genetic programming, which uses a grammar to control the structures being searched. A grammar is provided by the user as a template for rules. A set of rules is derived by using this grammar to form the initial population. Then, the main loop of GGP is entered. In each generation, individuals are selected stochastically to evolve offspring by the three genetic operators: crossover, mutation, and dropping condition. In each generation, the number of new individuals evolved equals the population size. Thus, the total number of individuals in the population is doubled. All individuals participate in the token competition and the replacement step to eliminate similar rules and increase diversity. One-half of the individuals with the higher fitness scores after token competition are retained and passed to the next generation. The whole process iterates until the maximum number of generations has been reached.

Grammar

The initial set of rules is created based on a grammar. The grammar of GGP governs the structures to be evolved, serving as a template for the rules. The initial population is created by randomly "filling" in this template. GGP will then search for the best set of rules without violating the grammar.

The grammar specifies that a rule is of the form "if *antecedents* then *consequent*." It specifies which attributes can appear in the antecedent part and which attributes can appear in the consequent part. It also specifies the descriptors of each attribute. The rule formats in various problems can be different. Thus, for each problem, a specific grammar is written so that the format of the rules can best fit the domain.

The use of grammar provides a powerful knowledge representation and allows great flexibility of the rule format. Rules with the user-desired structure can be learned, because the user can specify the required rule format using the grammar.

Genetic Operators

In rule learning using GGP, the search space is explored by generating new rules using three genetic operators. The genetic operators change the attribute descriptors in order to search for better rules.

Crossover produces one child from two parents, one designated as the primary parent and the other as the secondary parent. A part of the primary parent is selected and replaced by a compatible part from the secondary parent. The offspring produced must be valid according to the grammar.

Mutation is an asexual operation. A part in the parental rule is selected and replaced by a randomly generated part. The offspring has to be valid according to the grammar; thus, a selected part can only mutate to another part with a compatible structure.

The dropping condition [16] is a genetic operator for rule learning, to avoid subsumed rules. The rules evolved in GGP may be too restrictive and include redundant constraints. The dropping condition is used to generalize rules. A rule can be generalized if one descriptor in the antecedent part is dropped. The dropping condition randomly selects one attribute descriptor and then turns it into "any." That particular attribute is then no longer considered in the rule. For example, the parent:

if `attr1=0` and `attr2` between 100 150 and `attr3≠50`, then `attr4=T`.

may change to:

if `attr1=0` and `attr2` between 100 150 and *any*, then `attr4=T`.

Evaluation of Rules

An evaluation function based on the support-confidence framework [1] is developed as the fitness function in our rule-learning approach. *Support* measures the coverage of a rule. *Confidence factor* (cf) is the confidence of the consequent to be true under the condition that the antecedents are also true. For a rule "if A then B" and with a training set of N cases, support is $|A\&\ B| = N$ and the confidence factor is $|A\&\ B|/|A|$.

When evaluating the confidence of a rule, we need to consider the average probability of consequent (*prob*). The value *prob* is equal to $|B|/N$. We define *cf_part* as:

$$cf\_part = cf \times \log\left(\frac{cf}{prob}\right). \quad (1)$$

The log function measures the order of magnitude of the ratio *cf/prob*. A high value of *cf_part* requires simultaneously a high value of *cf* and a high value of the ratio *cf/prob*.

Support is another measure to be considered. If *support* is below a user-defined minimum threshold (*min_support*), the confidence factor of the rule is based on a small number of training examples, and we just ignore the confidence factor.

Our fitness function is defined to be:

$$\begin{aligned} &\textit{if } support < min\_support \textit{ then} \\ &\quad raw\_fitness = support \\ &\textit{else} \\ &\quad raw\_fitness = w1 \times support \\ &\quad + w2 \times cf\_part \end{aligned} \quad (2)$$

where the weights w_1 and w_2 are user defined and used to control the balance between the confidence and the support in learning. These two values have been set to one and eight, respectively, so that the confidence of the rule plays a more important role in the evaluation function.

Token Competition

The token competition [15] technique is employed in our rule-learning approach to search for a *set* of rules instead of just one rule. The concept is as follows. In the natural environment, once an organism has found a good place for living, it will try to exploit this niche and prevent interlopers from sharing the resources. The weaker interlopers individuals are hence forced to explore and find their own niches. In this way, the diversity of the population is in-

creased, so that healthy organisms are maintained in different niches.

Based on this mechanism, we assume that each record in the training set can provide a resource, called a token. If a rule can match a record, it sets a flag to indicate the token is seized. Other weaker rules, then, cannot get the token. The priority of receiving tokens is determined by the strength of the rules. A rule with a high score on *raw_fitness* [Eq. (2)] can exploit the niche by seizing as many tokens as it can. The other rules entering the same niche will have their strength decreased, because they cannot compete with the stronger rule. The fitness score of each individual is modified based on the tokens it can seize. The modified fitness is defined as:

$$modified\_fitness = raw\_fitness \times count / ideal \quad (3)$$

where *raw_fitness* is the fitness score obtained from the evaluation function, *count* is the number of tokens that the rule actually seized, and *ideal* is the total number of tokens that it can seize, which is equal to the number of records that the rule matches.

Learning Bayesian Networks from Discrete Variables

Besides learning rules from data, we have developed an approach to learn Bayesian network structures from discrete variables. The approach is based on evolutionary programming (EP) and the minimum description length (MDL) principle. The MDL principle has been applied to Bayesian network learning in our previous work [13]. The principle [18] states that the best model of a collection of data is the one that minimizes the sum of the encoding lengths of the data and the model itself. The MDL metric is defined in [12, 13] to measure the *total description length, DL,* of a network structure, *G*. The total description length of a network is the sum of description lengths of each node. This length of each node is defined based on two components, the *network description length* and the *data description length*. The former is the description length for encoding the network structure, which measures the simplicity of the network. The latter is the description length for encoding the data, which measures the accuracy of the network. For instance, consider two network structures, G_1 and G_2. Suppose G_1 has the structure, $N_1 \rightarrow N_3 \leftarrow N_2$, and G_2 has the structure, $N_1 \rightarrow N_3 \rightarrow N_2$. Suppose further that the data in the data set exhibit the following two dependency relationships: (1) the nodes N_1 and N_2 are independent; and (2) the node N_3 depends on both N_1 and N_2. The total description length of the structure G_1 will be less than that of the structure G_2, since G_1 fits more closely to the data.

To search for a good network structure, we developed an approach called MDLEP [14, 21], which uses evolutionary programming to optimize the MDL metric to learn the best Bayesian network structure. A Bayesian network is a DAG. A set of DAGs is randomly created to make up the initial population. Each DAG is evaluated by the MDL metric. Then, each DAG produces a child by performing a number of mutations. The child is also evaluated by using the MDL metric. The next generation of population is selected by tournaments among the parents and children. One-half of DAGs with the highest tournament scores are retained for the next generation. The process is repeated until the maximum number of generations is reached. The network with the lowest MDL score is output as the result.

Offspring in EP are produced by using a number of mutations. The probabilities of using 1, 2, 3, 4, 5, or 6 mutations are set to 0.2, 0.2, 0.2, 0.2, 0.1, and 0.1, respectively. These parameter values are selected to ensure that minor modifications of the offspring occur more frequently than do substantial variations. The mutation operators modify the edges of a DAG. If a cyclic graph is formed after the mutation, edges in the cycles are removed to keep it acyclic. The approach uses four mutation operators, which are designed to modify DAGs, with the same probabilities of being used:

1. Simple mutation randomly adds an edge between two nodes, or randomly deletes an existing edge from the parent.

2. Reversion mutation randomly selects an existing edge and reverses its direction.

3. Move mutation randomly selects an existing edge. It moves the parent of the edge to another node, or moves the child of the edge to another node.

4. Knowledge-guided mutation is similar to simple mutation, but the MDL scores of the edges guide the selection of the edge to be added or removed. The MDL metric of all possible edges in the network is computed before the learning algorithm starts. This mutation operator stochastically adds an edge with a small MDL metric to the parental network, or it deletes an existing edge with a large MDL metric.

Discretizing Continuous Variables while Learning Bayesian Networks

A Bayesian network can only represent discrete variables. One approach to handle databases with continuous variables is to discretize them first. The continuous variables are usually discretized by thresholds specified by a human. However, different discretization policies will produce different network structures. Causality will be lost if the discretization is not suitable. Thus, it is desirable to search for the best discretization policy before Bayesian networks are induced.

A *discretization sequence*, λ, defines a function that maps a continuous variable to a discrete variable. Each discretization sequence contains a list of threshold values. The variable will be discretized according to the range specified by the thresholds. A *discretization policy*, $\Lambda = \{\lambda_i : X_i \text{ is continuous}\}$, is a collection of discretization sequences for each continuous variable. This policy defines a new set of variables, $U^* = \{X_1^*, ..., X_n^*\}$, where $X_i^* = f_{\lambda_i}(X_i)$ if X_i is continuous and $X_i^* = X_i$ otherwise.

MDL for Discretization Policy

Friedman and Goldszmidt [6] extend the MDL score to evaluate the discretization policy while learning Bayesian network structures. The original training data, D, is discretized into a new data set, D^*. A Bayesian network structure, G, for the discretized variables, U^*, is learned from D^*. The new definition of the MDL score includes the description length of the network as well as description length of the discretization policy. This MDL score is composed of three parts. The first part is the description length of the network under the discretized data, as defined above. The second part is the length for encoding the particular discretization policy, Λ, over all of the possible discretization policies. The third part is the encoding length for reconstructing U from U^*.

Friedman and Goldszmidt have also described a greedy approach for learning discretization policy and Bayesian networks. This approach learns the discretization policy and network structures alternatively. It starts with an initial

discretization policy and learns Bayesian networks from the discretized data set by using the MDL metric. Based on the learned structure, a discretization policy is learned by using the MDL metric. In learning the discretization policy, only one variable is rediscretized at a time. The discretization sequence of this variable is reset to empty (i.e., no threshold values) first. The greedy approach searches for the "split" that gives the largest decrease in the MDL metric. The process is repeated until there is no improvement.

However, the greedy search algorithm can be easily trapped in a local optima. This approach also greatly depends on the initial settings. If the initial guess of discretization policy or network structure is not good, the result can be poor.

Learning Discretization Policy Using Genetic Algorithms

A GA is applied to optimize the new MDL metric, and thus the best network structure as well as the best discretization policy can be learned. It is less likely that the algorithm will be trapped in a local optima, because there is a population of individuals to explore the search space in parallel.

Our approach uses the iterative approach, as suggested in [6]. It starts with an initial discretization policy. MDLEP is then used to learn the best network structure. Based on this structure, a GA is used to learn the best discretization policy. The process is iterated until the maximum number of iterations is reached.

The GA starts with an initial randomly generated population. Each individual in the population is evaluated by the new MDL score defined in [6]. The good individuals are selected to produce offspring using the genetic operators. The offspring, in turn, produce the next generation until the maximum number of generations is reached.

Individual Representation

A discretization policy consists of discretization sequences for the continuous variables, and each discretization sequence consists of threshold values for discretization. We can limit the thresholds to midpoints between successive values that appear in the training data. Each individual should represent a possible discretization policy and should hence encode these threshold values.

We have used one bit string to represent one discretization sequence. The number of bits in each string equals the number of midpoints values of the variable (i.e., if variable i has s_i different values in the training data, the length of its bit string is $s_i - 1$). A "1" in the bit means the corresponding mid-point is included as a threshold in the discretization sequence. For example, in Fig. 1, the midpoints after the second, sixth, and ninth values are included in the discretization sequence of variable a.

To provide a more useful dis- cretization and simplify the computation, the user can limit the maximum number of thresholds appearing in the discretization sequence. Hence, the maximum number of "1"s in the bit string is limited. An individual stores the concatenation of the bit strings of each continuous variable, as shown in Fig. 2.

Genetic Operators

Four genetic operators are used. Other than the basic operators of reproduction, crossover, and mutation, another operator named "shift" is applied to evolve better discretization policies:

- Reproduction: The standard reproduction is used. The parent is selected and copied into the new generation.
- Crossover: The standard crossover can also be used. Two parents are selected. One random point in the bit string of the parents is selected as the crossover point. The bit string is cut into two parts at this point. The front parts of the two parents are exchanged to evolve two children.
- Mutation: A multiple-point mutation is used. For each variable, a bit is selected for mutation. There is a 50% chance that this bit is mutated. If mutation occurs, the bit is toggled.
- Shift: Shift is a special kind of mutation. This operator changes the threshold value in the discretization sequence to the next or previous midpoint value. One parent is selected for the shift operator. For each variable, a bit with a "1" is selected. There is a 50% chance that this bit is shifted. If a shift occurs, the bit is set to "0" and its neighbor bit (either left or right, with equal probability) is set to "1." Thus, the operator performs a local search for better threshold values.

Learning Results

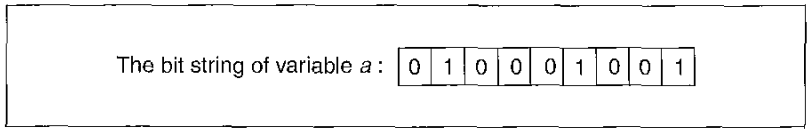
Fracture Database

Results of Bayesian Network Learning

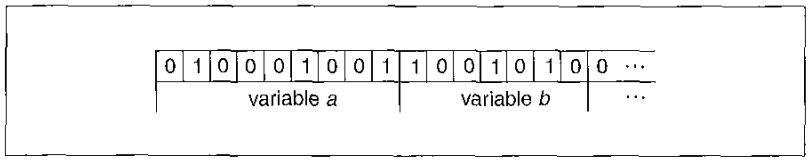
The relationships among the attributes are analyzed by learning a Bayesian network. We have used a population size of 50 for both MDLEP and GA. The discovered network structure is shown in Fig. 3. The discretization policy is shown in Table 3. *Day, Month, Weekday* and *Year* refer to different parts of the admission date. Age is divided into 0-4, 5-9, 10-12, and 13-16. The day and month are discretized into just one range, which means that they are not involved in any relationship in the Bayesian network. Year and stay are divided into three ranges.

From the network structure constructed, the following interesting relationships are observed:

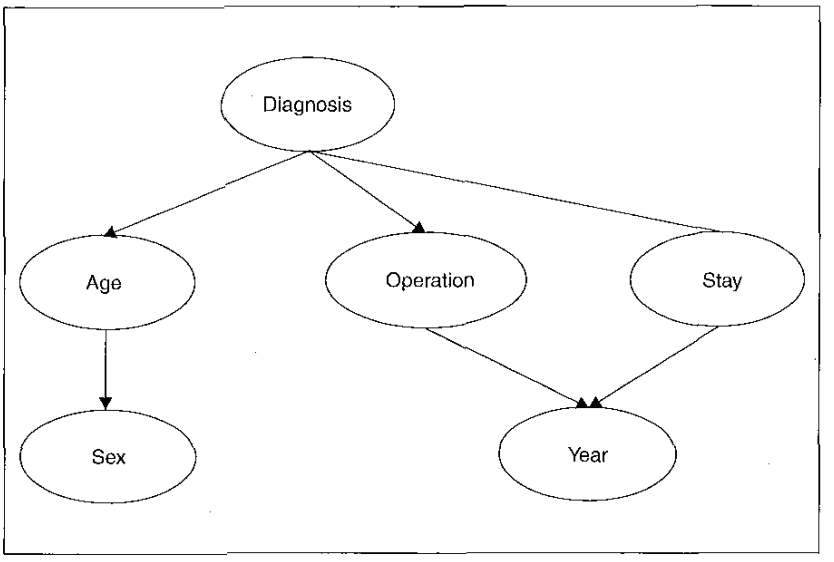
- The value of Diagnosis effects the values of Operation and Stay. Different fractures are treated with different operations and require different time for recovery.
- The value of `Diagnosis` effects the value of `Age`. Some fractures more frequently occur in particular age groups.
- The value of `Age` affects the value of `Sex`. It is observed that the young patients are more likely to be female, and older patients are more likely to be male.
- The value of `Operation` and `Stay` affects the value of `Year`. It is ob-



1. A bit string represents a discretization sequence.



2. The bit string in an individual.



3. The discovered network structure for the fracture database.

Table 3. Discretization Policy of the Fracture Database
Age: [0-4] [5-9] [10-12] [13-16]
Day: [1-31]
Month: [1-12]
Year: [1984-1987][1988-1991][1992-1996]
Stay: [0-3] [4-12] [13-1081]

served from the database that the length of stay in hospital is longer in the years 1985, 1986, and 1994, and open-reduction occurs more frequently for earlier years.

Results of Rule Learning

Based on the learned Bayesian network, we observe a causality model between diagnosis, operation, and stay. We wish to learn knowledge about these attributes. First, sex, age, and admission date are the possible causes of diagnosis. Second, these three attributes and diagnoses are the possible causes of operation and surgeon. Third, length of stay has all other attributes as possible causes. A grammar is written as a template for these three kinds of rules. This grammar is presented in Appendix 1. We have used a population size of 300 to run for 50 generations in the rule-learning step. The discovered rules are listed in Appendix 2.

The learning process can uncover knowledge about the age effect on fracture, the relationship between diagnosis and operation, and the effect of diagnosis and operation on length of staying in the hospital.

The results have been evaluated by medical experts. Previous analysis of fracture patterns only gave an overall injury pattern. Our system automatically uncovered relationships among different attribute values. The rules revealed some interesting patterns and rules that were not known before. The system can provide a good monitor of change of pattern if the data mining process is continued longitudinally over the years. It also provides the information for setting up a knowledge-based instruction system to help doctors in training.

Scoliosis Database

Results of Bayesian Network Learning

In this database, the attributes `Age`, `1stMCDeg`, `2ndMCDeg`, `Deg1` to `Deg4`, and `Mens` are continuous variables. For the attributes measuring degrees, the value 0 is a special value, as it means the curve does not exist. For `Mens`, the values –9 and 99 have special meanings, which indicate no menstruation. These values are specially handled such that they are always discretized from other values.

The learning of network structures and discretization policies are alternated for 20 iterations. For the learning of network structures using MDLEP, we have used a population of 50 to run for 100 generations. In each iteration of the learning of discretization policies using GAs, the population size is 50 and the number of generations is 10. The discovered Bayesian network structure learned from this data set is shown in Fig. 4. The discretization policy is shown in Table 4. The age is divided into 0-12 (child), 13-16 (adolescence), 17-21, and over 22. The degrees and `Mens` are divided into different ranges.

The discovered Bayesian network shows some physical relationships among attributes. For example, the network shows that `1stMCDeg` and `2ndMCDeg` are related with `Deg2`. The two major curves are defined as those with the largest degrees among the four curves, and most likely `Deg2` is involved. `Deg1` and `Deg2` can imply `Deg4` and `Deg3`, because if the degree of first or second curves are small, the degrees of the remaining curves are either zero or small. The network also reveals some obvious patterns; `Age` affects `Mens` and `RI` (the maturity), and the value of `Mens` affects `Sex`. In addition, the following relationships are observed:

- The value of `Operation` affects the value of `1stMCDeg`. If `Operation` is equal to observation, the value `1stMCDeg` is smaller. If `Operation` equals to surgery, the value of `1stMCDeg` is large.
- The value of `Deg3` affects the value of `1stCurveT1`. If `Deg3` is large, the spine has three or more curves, and most likely the first curve starts at the first vertebra T1.
- The value of `Deg3` affects the value of `TSIDir`. If `Deg3` is small, most of the time the direction of trunk shift is null
- The value of `Treatment` affects the value of `1stMCDeg`. If treatment is bracing, most likely the degree of the first major curve is small. In contrast, if operation is needed, the degree of the first major curve is usually large.

Results of Rule Learning

Medical experts are interested in inducing knowledge about classification of scoliosis. Scoliosis can be classified as Kings, thoracolumbar (TL), and lumbar (L), while Kings can be further subdivided into K-I, II, III, IV, and V. This domain knowledge has been incorporated into the design of the rule grammar.

The population size used in the rule-learning step is 100 and the maximum number of generations is 50. For

each class of scoliosis, a number of rules are obtained. The discovered rules are listed in Appendix 3.

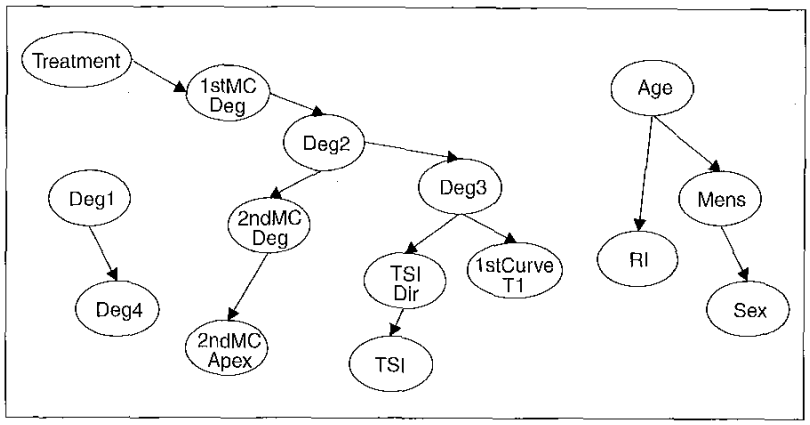
The rules discovered are generally consistent with the knowledge of medical experts. However, there is an unexpected rule for the classification of King-II. Under the conditions specified in the antecedents, our system found a rule with a confidence factor of 52% that the classification is King-II. However, the domain expert suggests the class should be King-V! After an analysis of the database, we found that serious data errors existed in the current database and that some records contained an incorrect scoliosis classification. The rules for TL and L also showed something different in comparison with the rules suggested by the clinicians. According to our rules, the classification always depends on the location of the *first major curve*, while according *to the domain expert, the classification always depends on the larger major curve*. After discussion with the domain expert, it was agreed that the existing rules were not defined clearly enough and that our rules were more accurate than theirs. Thus, our rules provided hints to the clinicians to reformulate their concepts.

The largest effect on the clinicians from the data mining analysis of the scoliosis database is the fact that many rules set out in the clinical practice are not clearly defined. The usual clinical interpretation depends on subjective experience. Our data mining effort revealed quite a number of mismatches in the classification on the type of Kings curves. After a careful review by the senior surgeon, it appears that the database entries by junior surgeons may not be accurate and that the data mining rules discovered are in fact more accurate! The classification rules must, therefore, be quantified. These rules can help in the training of doctors and act as an intelligent means to validate and evaluate the accuracy of the clinical database.

Conclusion

We have presented our approach for knowledge discovery from two specific medical databases. First, rules are learned to represent the interesting patterns of the data. Second, Bayesian networks are induced to act as causality relationship models among the attributes. The Bayesian network learning process is divided into two phases. In the first phase, a discretization policy is learned to discretize the continuous variables, and then Bayesian network structures are induced in the second phase. We employ advanced evolutionary algorithms such as generic genetic programming, evolutionary programming, and genetic algorithms to conduct the learning tasks.

From the fracture database, we discovered knowledge about the patterns of child fractures. From the scoliosis database, we discovered knowledge about the classification of scoliosis. We also found unexpected rules that led to discovery of errors in the database. These results demonstrate that the knowledge discovery process can find interesting knowledge about the data, which can provide novel clinical knowledge as well as suggest refinements of the existing knowledge.



4. The discovered network structure for the scoliosis database.

Table 4. Discretization Policy of the Scoliosis Database
`Age`: [0-12] [13-16] [17-21] [22-41]
`1stMCDeg`: [5-13] [14-29] [30-35] [36-52] [53-112]
`2ndMCDeg`: [0-0] [5-23] [24-36] [37-65]
`Deg1`: [3-11] [12-35] [36-52] [54-112]
`Deg2`: [0-0] [2-26] [27-36] [37-52] [53-93]
`Deg3`: [0-0][3-21] [22-60]
`Deg4`: [0-0][13-34]
`Mens`: [-9 - -9] [0-4] [5-30] [99-99]

Acknowledgment

This research was partially supported by the RGC Earmarked Grant CUHK 4161/97E of Hong Kong and the CUHK Engineering Faculty Direct Grant 2050179.

Man Leung Wong is an assistant professor at the Department of Information Systems at Lingnan University, Tuen Mun, Hong Kong. His research interests are knowledge discovery in databases, machine learning, electronic commerce, evolutionary computation, knowledge acquisition, fuzzy logic, and approximate reasoning. He received his B.Sc., M.Phil., and Ph.D. in computer science from the Chinese University of Hong Kong in 1988, 1990, 1995, respectively. He is a member of the IEEE and the ACM.

Wai Lam received a Ph.D. in computer science from the University of Waterloo, Canada, in 1994. He worked as a visiting research associate at Indiana University-Purdue University in Indianapolis in 1995 and as a postdoctoral fellow at the Distributed Adaptive Search Laboratory of the University of Iowa in 1996. He is currently an assistant professor in the Department of Systems Engineering and Engineering Management of the Chinese University of Hong Kong. His current interests include data mining, intelligent in-

formation retrieval, machine learning, reasoning under uncertainty, and digital libraries.

Kwong Sak Leung is currently a professor and chairman of the Department of Computer Science and Engineering of the Chinese University of Hong Kong and was appointed the head of Graduate Division of Computer Science between April 1992 and July 1997. He worked as a senior engineer and system analyst at ERA Technology and the headquarters computer center of the Central Electricity Generating Board, respectively, in England for five years before joining Chinese University in August 1985. Dr. Leung received his B.Sc. and Ph.D. degrees from the University of London in 1977 and 1980, respectively, and is a member of IEE and ACM, a senior member of IEEE, and a chartered engineer. He was one of the founding members and the chairman of ACM Hong Kong Chapter and a council member of the Hong Kong Computer Society. He is serving as a member of the Engineering Panel of the Research Grant Council of the University Grant Committee of Hong Kong. Dr. Leung is a member of editorial board for *Fuzzy Sets and Systems* and *IT Magazine*. He has served as chairman and member of numerous international conference organizing and program committees. He has authored and co-authored over 130 publications. His research interests are in the areas of knowledge engineering, expert systems, genetic algorithms and programming, data mining and knowledge acquisition, Chinese processing, fuzzy logic applications, and AI architecture.

Po Shun Ngan received his B.Eng. degree in computer engineering and M.Phil. degree in computer science and engineering from the Chinese University of Hong Kong, respectively, in 1996 and 1998. His research interests include data mining, evolutionary computation, grammar-based genetic programming, and machine learning.

Jack C.Y. Cheng graduated from the Medical Faculty of the University of Hong Kong in 1976 with an M.B.B.S. He then completed the specialist training in orthopaedic surgery in Hong Kong, Oxford, and Edinburgh and obtained the orthopaedic specialist qualification (FRCS Ed Orth) in 1984. With further subspecialization in children orthopaedics, he was involved in the treatment of large numbers of children fractures and other congenital diseases. He is currently the chairman of Department of Orthopaedics and Traumatology of the Chinese University of Hong Kong and head of the Paediatric Orthopaedic Service of the Prince of Wales Hospital. His interest in information technology applications in medical fields both in teaching and research has brought him into close liaison with many computer specialists. The application of data mining in medical fields as presented in this article has opened up new areas of interesting research. His main areas of current research include studies of children fractures, biomaterials, scoliosis, and bone mineral profiles in normal and pathological conditions.

Address for Correspondence: Wai Lam, Deptartment of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong. E-mail: wlam@se.chuk.edu.hk

References

1. **Agrawal R, Imielinski T, and Swami A:** Mining association rules between sets of items in large databases. In: *Proc 1993 Int Conf Management of Data (SIGMOD 93)*, pp. 207-216, 1993.

2. **Charniak E:** Bayesian networks without tears. *AI Magazine* 12(4): 50-63, 1991.

3. **Fayyad UM, Piatesky-Shapiro G, and Smyth P:** From data mining to knowledge discovery : An overview. *AI Magazine* 17(3): 37-54, Fall 1996.

4. **Fogel DB:** An introduction to simulated evolutionary optimization. *IEEE Trans Neural Networks*, 5: 3-14, 1994.

5. **Fogel L, Owens A, and Walsh M:** *Artificial Intelligence through Simulated Evolution.* New York: Wiley, 1966.

6. **Friedman N and Goldszmidt M:** Discretizing continuous attributes while learning Bayesian networks. In: *Proc Int Conf Machine Learning*, pp. 157-165, 1996.

7. **Goldberg DE:** *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA: Addison-Wesley, 1989.

8. **Heckerman D and Wellman MP:** Bayesian networks. *Communications of the ACM*, 38(3): 27-30, March 1995.

9. **Holland JH:** *Adaptation in Natural and Artificial Systems.* 2nd Ed. Cambridge, MA: Bradford/MIT Press, 1992.

10. **Koza JR:** *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: Bradford/MIT Press, 1992.

11. **Koza JR:** *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, MA: Bradford/MIT Press, 1994.

12. **Lam W:** Bayesian network refinement via machine learning approach. *IEEE Trans Pattern Anal and Machine Intell* 20(3): 240-251, 1998.

13. **Lam W and Bacchus F:** Learning Bayesian belief networks - An approach based on the MDL principle. *Computational Intelligence*, 10(3): 269-293, 1994.

14. **Lam W, Wong ML, Leung KS, and Ngan PS:** Discovering probabilistic knowledge from databases using evolutionary computation and minimum description length principle. In: *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pp. 786-794, 1998.

15. **Leung KS, Leung Y, So L, Yam KF:** Rule learning in expert systems using genetic algorithm: 1, concepts. In: *Proc 2nd Int Conf Fuzzy Logic and Neural Networks (Iizuka, Japan)*, pp. 201-204, 1992.

16. **Michalski RS:** A theory and methodology of inductive learning. In Michalski RS, Carbonell JG, and Mitchell TN (Eds): *Machine Learning: An Artificial Intelligence Approach.* Los Altos, CA: Tioga, 1983.

17. **Rechenberg I:** *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (Evolution Strategy: Optimization of Technical Systems by Means of Biological Evolution).* Stuttgart: Fromman-Holzboog, 1973.

18. **Rissanen J:** Modeling by shortest data description. *Automatica* pp. 465-471, 1978.

19. **Schwefel HP:** *Numerical Optimization of Computer Models.* Chichester, UK: Wiley, 1981.

20. **Wong ML:** Evolutionary program induction directed by logic grammars. PhD dissertation, The Chinese University of Hong Kong, 1995.

21. **Wong ML, Lam W, and Leung KS:** Using evolutionary computation and minimum description length principle for data mining of probabilistic knowledge. *IEEE Trans Pattern Anal and Machine Intell* 21(2): 174-178, 1999.

22. **Wong ML and Leung KS:** Inducing logic programs with genetic algorithms: The genetic logic programming system. *IEEE Expert* 10(5): 68-76, 1995.

23. **Wong ML, and Leung KS:** Evolutionary program induction directed by logic grammars. *Evolutionary Computation*, 5: 143-180, 1997.

Appendix 1. The Grammar for the Fracture Database

This grammar is not completely listed. The grammar for the other attribute descriptors is similar to the part of the grammar in lines 11-19.

1. *Rule* → *Rule1* | *Rule2* | *Rule3*
2. *Rule1* → if *Antes1*, then *Consq1*
3. *Rule2* → if *Antes1* and *Antes2*, then *Consq2*
4. *Rule3* → if *Antes1* and *Antes2* and *Antes3, then Consq2*
5. *Antes1* → *Sex1* and *Age1* and *Admday1*
6. *Antes2* → *Diagnosis1*
7. *Antes3* → *Operation1 and Surgeon1*
8. *Consq1* → *Diagnosis_descriptor*
9. *Consq2* → *Operation_descriptor* | *Surgeon_descriptor*
10. *Consq3* → *Stay_descriptor*
11. *Sex1* → any | *Sex_descriptor*
12. *Sex_descriptor* → sex = sex_const
13. *Admday1* → any | *Admday_descriptor*
14. *Admday_descriptor* → admday_day between day_const day_const
15. *Admday_descriptor* → admday_month between month_const month_const
16. *Admday_descriptor* → admday_year between year_const year_const
17. *Admday_descriptor* → admday_weekday between weekday_const weekday_const
18. *Diagnosis1* → any | *Diagnosis_descriptor*
19. *Diagnosis_descriptor* → diagnosis is diagnosis_const

Appendix 2. The Best Rule Set Learned from the Fracture Database

Type I Rules: About Diagnosis

1. Humerus
if age is between 2 and 5, then diagnosis is Humerus.
Fitness: 3.48
Confidence: 39.75%; Support: 8.42%; Probability of consequent: 23.43%

2. Radius
if sex is M, and age is between 11 and 13, then diagnosis is Radius.
Fitness: 3.04 Confidence: 51.43%; Support: 10.01%; Probability of consequent: 36.10%

Type II Rules: About Operation/Surgeon

1. Radius versus CR+POP
if age is between 0 and 7, and admission year between 1988 and 1993, and diagnosis is Radius, then operation is CR+POP.
Fitness: 8.56
Confidence: 50.61%; Support: 3.19%; Probability of consequent: 17.72%

2. Tibia versus No Operation
if age is between 1 and 7, and diagnosis is Tibia, then operation is Null (i.e. no operation).
Fitness: 7.86
Confidence: 74.05%; Support: 3.78%; Probability of consequent: 38.11%

3. Ulna versus CR+POP
if age is between 1 and 12, and admission year between 1989 and 1992, and diagnosis is Ulna, then operation is CR+POP.
Fitness: 7.19
Confidence: 47.37%; Support: 3.50%; Probability of consequent: 17.72%
if diagnosis is Ulna, then operation is CR+POP. Fitness: 4.23 Confidence: 36.17%; Support: 7.40%; Probability of consequent: 17.72%

4. Radius versus CR+K-Wire
if admission year is between 1992 and 1994, and diagnosis is Radius, then operation is CR+K-Wire.
Fitness: 4.10
Confidence: 34.03%; Support: 3.83%; Probability of consequent: 16.23%

5. Humerus versus CR+K-Wire
if diagnosis is Humerus, then operation is CR+K-Wire.
Fitness: 2.52
Confidence: 27.96%; Support: 6.06%; Probability of consequent: 16.23%

6. Ulna versus OR
if age is between 11 and 15, and diagnosis is Ulna, then operation is OR. Fitness: 3.24 Confidence: 33.20%; Support: 3.25%; Probability of consequent: 18.26%

7. Age versus OR
if sex is M, and age is between 13 and 17, and admission year between 1985 and 1989, then operation is OR.

Fitness: 2.57
Confidence: 30.53%; Support: 3.22%; Probability of consequent: 18.26%

8. Age versus No Operation
if age is between 0 and 7, then operation is Null (i.e. no operation).
Fitness: 1.08
Confidence: 43.33%; Support: 16.22%; Probability of consequent: 38.11%

Type III Rules: About Stay

1. Femur versus Stay
if admission year between 1985 and 1996, and diagnosis is Femur, then stay is between 8 and 2000 days. (i.e. stay 8 days or more, since 2000 is the maximum value of stay)
Fitness: 21.99
Confidence: 70.87%; Support: 3.14%; Probability of consequent: 10.24%
if diagnosis is Femur, then stay is between 5 and 2000 days. (i.e. stay 5 days or more)
Fitness: 18.70
Confidence: 80.99%; Support: 3.30%; Probability of consequent: 19.22%

2. Tibia versus Stay
if age between 5 and 12, and diagnosis is Tibia, then stay is between 3 and 2000. (i.e. stay 3 days or more)
Fitness: 8.93
Confidence: 78.92%; Support: 5.05%; Probability of consequent: 39.15%

3. OR versus Stay
if age between 2 and 14, and diagnosis is Humerus, and operation is OR, then stay is between 3 and 25 days.
Fitness: 8.86
Confidence: 75.57%; Support: 3.52%; Probability of consequent: 36.51%
if admission is between 1985 and 1987, and operation is OR, then stay is between 3 and 10 days.
Fitness: 6.99
Confidence: 65.52%; Support: 3.47%; Probability of consequent: 33.85%
if operation is OR, then stay is between 3 and 25 days.
Fitness: 6.13
Confidence: 64.90%; Support: 12.22%; Probability of consequent: 36.51%

4. No operation versus Stay
if age is between 10 and 14, and admission year is between 1987 and 1996, and diagnosis is Radius, and operation is Null, then stay is between 0 and 1 day.
Fitness: 9.55
Confidence: 77.00%; Support: 3.09%; Probability of consequent: 35.65%
if operation is Null, then stay is between 0 and 1 day.
Fitness: 3.38
Confidence: 52.06%; Support: 19.62%; Probability of consequent: 35.65%

5. Radius versus Stay
if age between 6 and 12, and admission year is between 1989 and 1992, and diagnosis is Radius, and operation is CR+POP, then stay is between 1 and 2 days.
Fitness: 6.01
Confidence: 81.11%; Support: 3.22%; Probability of consequent: 51.29% if diagnosis is Radius, and operation is CR+POP, then stay is between 1 and 2 days.
Fitness: 5.49
Confidence: 78.57%; Support: 10.22%; Probability of consequent: 51.29%
if age is between 0 and 8, and diagnosis is Radius, then stay is between 0 and 3 days.
Fitness: 2.89
Confidence: 86.92%; Support: 10.19%; Probability of consequent: 71.30%

6. Humerus versus Stay
if diagnosis is Humerus, and operation is CR+K-WIRE, then stay is between 2 and 5 days.
Fitness: 3.90
Confidence: 67.30%; Support: 4.56%; Probability of consequent: 47.16%

7. Year versus Stay
if admission year is between 1985 and 1987, then stay is between 3 and 10 days.
Fitness: 2.58
Confidence: 46.98%; Support: 8.65%; Probability of consequent: 33.85%

Appendix 3. The Best Rule Set Learned from the Scoliosis Database

Rules for Classification

King-I
1. if 1stMCGreater=N and 1stMCApex=T1-T8 and 2ndMCApex=L3-L4, then King-I.
Fitness: 20.20
Confidence: 100%; Support: 0.86%; Probability of consequent: 28.33%

2. if 1stMCGreater=N and 1stMCDeg=21-80 and 1stMCApex =T1-T12 and 2ndMCApex=L2-L3, then King-I.
Fitness: 19.06
Confidence: 96.67%; Support: 6.22%; Probability of consequent: 28.33%

3. if 1stMCGreater=N and L4Tilt=Y and 1stMCApex =T1-T10 and 2ndMCApex=L2-L5, then King-I.
Fitness: 18.92

Confidence: 96.15%; Support: 10.73%; Probability of consequent: 28.33%

King-II

1. if 1stCurveT1=N and 1stMCGreater=Y and 1stMCDeg=16-45 and 2ndMCDeg=28-54 and 1stMCApex =T4-T11 and 2ndMCApex=L2-L3, then King-II.
Fitness: 16.63
Confidence: 100.00%; Support: 1.07%; Probability of consequent: 35.41%

2. if 1stMCGreater=Y and L4Tilt=Y and 1stMCDeg=22-77 and 2ndMCDeg=19-54 and 1stMCApex=T1-T11 and 2ndMCApex=L2-L2, then King-II.
Fitness: 12.85
Confidence: 87.88%; Support: 6.22%; Probability of consequent: 35.41%

3. if 1stMCGreater=Y and L4Tilt=Y and 1stMCApex=T6-T10 and 2ndMCApex=L2-L5, then King-II.
Fitness: 10.52
Confidence: 79.76%; Support: 14.38%; Probability of consequent: 35.41%

4. if 1stMajorCurveGreater=Y and 2ndMCDeg=8-95 and 1stMCApex=T3-T11 and 2ndMCApex= T4-T10, then King-II.
Fitness: 3.32
Confidence: 52.17%; Support: 7.73%; Probability of consequent: 35.41%

King-III

1. if 1stCurveT1=N and L4Tilt=N and 1stMCApex=T1-T9 and 2ndMCApex=Null, then King-III.
Fitness: 5.87
Confidence: 25.87%; Support: 0.86%; Probability of consequent: 7.94%

2. if L4Tilt=N and 1stMCApex=T2-T6 and 2ndMCApex=T2-T11, then King III.
Fitness: 4.86
Confidence: 25.71%; Support: 1.93%; Probability of consequent: 7.94%

King-IV

1. if 1stCurveT1=Y and 1stMCGreater=Y and L4Tilt=Y and 1stMCApex=L5-T10 and 2ndMCApex=T9-L5, then King-IV.
Fitness: 11.10
Confidence: 29.41%; Support: 1.07%; Probability of consequent: 2.79%

2. if 1stMCGreater=Y and L4Tilt=Y and 1stMCApex=T10-L5 and 2ndMCApex=T5-L4, then King-IV.
Fitness: 6.02
Confidence: 19.35%; Support: 1.29%; Probability of consequent: 2.79%

King-V

1. if 1stMCGreater=Y and L4Tilt=Y and 1stMCApex=T2-T5 and 2ndMCApex=T9-T11, then King-V.
Fitness: 22.75
Confidence: 62.50%; Support: 1.07%; Probability of consequent: 6.44%

2. if 1stMCGreater=N and 2ndMCDeg=37-70 and 1stMCApex=T4-T7 and 2ndMCApex=T2-T11, then King-V.
Fitness: 19.98
Confidence: 57.14%; Support: 0.86%; Probability of consequent: 6.44%

3. if 1stCurveT1=Y and 1stMCGreater=Y and L4Tilt=Y and 1stMCDeg=3-35 and 1stMCApex=T2-T6 and 2ndMCApex=T7-T9, then King-V.
Fitness: 16.42
Confidence: 50.00%; Support: 0.86%; Probability of consequent: 6.44%

TL

1. if 1stMCGreater=Y and 1stMCApex=T11-T12 and 2ndMCApex=Null, then TL.
Fitness: 19.49
Confidence: 41.18%; Support: 1.50%; Probability of consequent: 2.15%

L

1. if 1stMCGreater=Y and L4Tilt=N and 1stMCApex=L2-L5 and 2ndMCApex=Null, then L.
Fitness: 26.32
Confidence: 62.50%; Support: 1.07%; Probability of consequent: 4.51%

2. if 1stCurveT1=N and L4Tilt=N and 2ndMCDeg=Null and 1stMCApex=L1-L3 and 2ndMCApex=Null, then L.
Fitness: 21.59
Confidence: 54.17%; Support: 2.79%; Probability of consequent: 4.51%

3. if 1stCurveT1=N and 1stMCApex=L2-L5 and 2ndMCApex=Null, then L.
Fitness: 16.84
Confidence: 45.45%; Support: 2.15%; Probability of consequent: 4.51%