SETIT 2007
4rth International Conference: **S**ciences of **E**lectronic, **T**echnologies of **I**nformation and **T**elecommunications
March 25-29, 2007 – TUNISIA

Knowledge Discovery in Data Mining Using Fuzzy c-Means Model and Genetic Programming

Mahdi A. Salman 1*, Samaher Hussein Ali 2**

* *Department of Computer Science, University of Babylon, Iraq*
mahdi@thurayanet. com

** *Department of Computer Science, University of Babylon, Iraq*
Samaher_hussein@yahoo.com

Abstract: This paper presents a methodology for discovering classification rules in data mining. The attributes defining the data space can be inadequate, making it difficult to discover high-quality knowledge. In order to solve this problem, this paper proposes a fuzzy c-means model (FCM) for attribute clustering after preprocessing of that attributes (features). The Genetic Programming (GP) is used to determine which such features are the most predictive. Then compare these rules with all the clusters and add the rules which success 80% to knowledge base. Using five well known datasets held at the UCI repository [1] .

Key words: extraction rules, fuzzy c-means algorithm, genetic programming, knowledge discovery.

INTRODUCTION

Data mining is the process of extraction knowledge hidden from large volumes of raw data. In other word data mining automates the process of finding relationships and patterns in a row data and delivers results that can be either utilized in an automated decision support system or assessed by human analyst.

Data mining consists of the (semi-)automatic extraction of knowledge from data. This statement raises the question of what kind of knowledge we should try to discover. Although this is subjective issue, we can mention three general properties that the discovered knowledge should satisfy; namely, it should be accurate, comprehensible, and interesting.

In data mining we are often interested in discovering knowledge which has a certain predictive power. The basic idea is to construct new attributes out of the original ones, transforming the original data representation into a new one where regularities in the data are more easily detected by the classification algorithm, which tends to improve the predictive accuracy of the latter.

Attribute construction methods can be roughly divided into two groups, with respect to the construction strategy: hypothesis-driven methods and data-driven methods [1].

[1]http://www.ics.uci.edu/~mlearn/MLRepository.html

Hypothesis-driven methods construct new attributes out of previously-generated hypotheses (discovered rules or another kind of knowledge representation). In general they start by constructing a hypothesis, for instance a decision tree, and then examine that hypothesis to construct new attributes [2].

By contrast, data-driven methods do not suffer from the problem of depending on the quality of previous hypotheses. They construct new attributes by directly detecting relationships in the data.

The process of attribute construction can also be roughly divided into two approaches, namely the interleaving approach and the preprocessing approach. In the preprocessing approach the process of attribute construction is independent of the inductive learning algorithm that will be used to extract knowledge from the data. In other words, the quality of a candidate new attribute is evaluated by directly accessing the data, without running any inductive learning algorithm. In this approach the attribute construction method performs a preprocessing of the data, and the new constructed attributes can be given to different kinds of inductive learning methods.

By contrast, in the interleaving approach the process of attribute construction is intertwined with the inductive learning algorithm. The quality of a candidate new attribute is evaluated by running the inductive learning algorithm used to extract

knowledge from the data, so that in principle the constructed attributes' usefulness tends to be limited to that inductive learning algorithm. An example of an attribute construction method following the interleaving approach can be found in [3].

In this paper we follow the data-driven strategy and the preprocessing approach, mainly for two reasons**.** First, using this combination of strategy/approach the constructed attributes have a more generic usefulness, since they can help to improve the predictive accuracy of any kind of inductive learning algorithm. Second, an attribute construction method following the preprocessing approach tends to be more efficient than its interleaving counterpart, since the latter requires many executions of an inductive learning algorithm.

1. Knowledge Discovery in Database

The process of knowledge extraction from database combines methods and statistical tools, machine learning and databases to find a mathematical and/or logical description, which can be eventually complex, of patterns and regularities in data [4].

The knowledge extraction from a large amount of data should be seen as an interactive and iterative process, and not as a system of automatic analysis. In this way, we cannot simply expect an extraction of useful knowledge by submitting a group of data to a "black box".

The interactivity of the knowledge discovery in database (KDD) process refers to the greater understanding, on the part of the users of the process, of the application domain. This understanding involves the selection of a representative data subset, appropriate pattern classes and good approaches to evaluating the knowledge. For a better understanding the functions of the users that the KDD process, users are divided in three classes: (a) Domain Expert, who should process a large understanding of the application domain; (b) Analyst, who executes the KDD process and therefore, he should have a lot of knowledge of the stages that compose this process and (c) Final User, who does not need to have much knowledge of the domain. Frequently, the Final User uses knowledge extracted from the KDD process to aid him in a decision– making process [5].

1.1. The Steps in the KDD Process

Even through the KDD processes have merged from different fields it has almost the same steps in all of the different approaches. These steps are [6]:

(a) Developing an understanding of the application domain, the relevant prior knowledge, and the goal(s) of the end-user.

(b) Creating a target data set: selection a data set or focusing on a subset of variables or data samples, on which discovery is to be performed.

(c) Data cleaning and preprocessing : basic operation such as removing noise or model for noise, deciding on strategies for handling missing data fields, accounting for time sequence information and known changes.

(d) Data reduction: preparing the data set, removing some attributes to suit the set to the goal.

(e) Choosing the data mining task: deciding whether the goal for the KDD process is classification, regression, clustering, ect.

(f) Choosing the data mining algorithm(s): selection method(s) to be used for searching for pattern in the data. This also includes deciding which models and parameters may be appropriate.

(g) Data mining: search for patterns of interest in a particular representational form or set of forms: classification rules or tree, regression or clustering.

(h) Interpreting mined patterns, possible return to any of the steps a-g for further iteration.

(i) Consolidating discovered knowledge.

2. Preprocessing

We need to preprocess data since: (1) Data quality is a key issue with data mining. (2) To increase the accuracy of the mining has to perform data preprocessing. The researchers in data mining fields find that 80% of mining efforts often spend their time on data quality. So. How to preprocess data? This done through: Data Clustering, Data Integration, Data Reduction, Data Normalization.

2.1. Fuzzy c-Means Algorithm

Partitional clustering essential deals with the task of partitioning a set of entities into a number of homogeneous clusters, with respect to a suitable similarity measure. Due to the fuzzy clustering methods have been developed following the general fuzzy set theory strategies outlined by Zadeh [7]. The main difference between the traditional hard clustering and fuzzy clustering can be stated as follows. While in hard clustering an entity belongs only to one clusters, in fuzzy clustering entities are allowed to belong to many clusters with different degrees of membership.

The fuzzy c-means algorithm [8] is one of the most widely used methods in fuzzy clustering. It is based on the concept of fuzzy c-partition, introduced by Ruspini [9], summarized as follows.

Let $X=\{x_{1,\ldots,}x_n\}$ be a set of given data, where each data point x_k (k=1,…,n) is a vector in $\mathcal{R}^P$, U_{cn} be a set of real c×n matrices, and c be an integer, $2\leq c<n$. Then, the fuzzy c-partition space for X is the set

$$M_{fcn}= \{U \in U_{cn} : u_{ik} \in [0,1] \quad (1)$$

$$\sum_{i=1}^{c} u_{ik} = 1, \quad 0 < \sum_{k=1}^{n} u_{ik} < n\}$$

Where u_{ik} is the membership value of x_k in cluster i (i=1,…,c).

The aim of the FCM algorithm is to find an optimal fuzzy c-partition and corresponding prototypes minimizing the objective function

$$J_m(U,V;X) = \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m \|x_k - v_i\|^2 \qquad (2)$$

In (2), V=(v1,v2,…,vc) is a matrix of unknown cluster centers(prototypes) $v_i \in \mathcal{R}^p$, $\|.\|$ is the Euclidean norm, and the weighting exponent m in [1,∞) is a constant that influences the membership values.

To minimize criterion J_m, under the fuzzy constraints define in (1), the FCM algorithm is define as an alternating minimization algorithm (cf.[3] for the derivations), as follows.

Choose a value for c,m and ε , a small positive constant; then, generate randomly a fuzzy c-partition U^0 and set iteration number t=0. A two-step iterative process works as follows. Given the membership values $u_{ik}^{(t)}$, the cluster centers $v_i^{(t)}$,(i=1,…,c) are calculated by

$$V_i^{(t)} = \frac{\sum_{k=1}^{n} (u_{ik}^{(t)})^m . x_k}{\sum_{k=1}^{n} (u_{ik}^{(t)})^m} \qquad (3)$$

Given the new cluster centers $v_i^{(t)}$, update membership values $u_{ik}^{(t)}$:

$$u_{ik}^{(t+1)} = \left[\sum_{j=1}^{c} \left(\frac{\|x_k - v_i^{(t)}\|^2}{\|x_k - v_j^{(t)}\|^2} \right)^{\frac{2}{m-1}} \right]^{-1} \qquad (4)$$

The process stops when $|U^{(t+1)} - U^{(t)}| \le \varepsilon$, or a predefined number of iterations is reached.

Several algorithms for clustering data when the number of clusters is known a priori are available in the literature viz., K-means, branch and bound procedure, maximum likelihood estimate technique, graph theoretic approaches.

In most real life situations the number of clusters in a data set is not known a priori. The real challenge in this situation is to be able to automatically evolve a proper value of C as well as providing the appropriate clustering. In this article, we propose a FCM based clustering technique which can automatically evolve the appropriate clustering of a data set.

2.2. Fuzzy C-Means Clustering Algorithms

In this section, an attempt has been made to use fuzzy c-means clustering algorithms for automatically clustering a data set. This includes determination the best seed for each cluster as well as appropriate clustering of the data.

Procedure Fuzzy C-Means

Input: data set is array of attributes**.**

Output: set of cluster centers.

Preparations:

1:- Fix c, $2 \le c < d$

2:- Choose any inner product norm metric for Rn.

3:- Choose the termination tolerance $\delta > 0$,

e.g between 0.01 and 0.001.

4:- Fix w, $1 \le w < \infty$, e.g. 2.

5:- Initialise U (0) $\in Mfc$, (e.g. randomly).

Repeat

Step 1: Compute cluster prototypes:

$$\mathbf{c}_l^{(i)} = \frac{\sum_{j=1}^{d} \left(u_{ij}^{(l-1)}\right)^w \mathbf{m}_j}{\sum_{j=1}^{d} \left(u_{ij}^{(l-1)}\right)^w}, \quad 1 \le i \le c$$

Step 2: Compute distances:

For all clusters $1 \le i \le c$,

For all data objects $1 \le j \le d$,

$$d_{\mathbf{A}}^2\left(\mathbf{m}_j, \mathbf{c}_l^{(i)}\right) = \left(\mathbf{c}_l^{(i)} - \mathbf{m}_j\right)^T \mathbf{A} \left(\mathbf{c}_l^{(i)} - \mathbf{m}_j\right)$$

Step 3: Update the partition matrix:

If $d_{\mathbf{A}}\left(\mathbf{m}_j, \mathbf{c}_l^{(i)}\right) > 0$ for $1 \le i \le c, 1 \le j \le d$,

$$u_{ij}^{(l)} = \frac{1}{\sum_{k=1}^{c} \left(d_{\mathbf{A}}^2(\mathbf{m}_j, \mathbf{c}^{(i)}) / d_{\mathbf{A}}^2(\mathbf{m}_j, \mathbf{c}^{(k)})\right)^{1/(w-1)}}$$

otherwise

$$u_{ij}^{(l)} = 0 \text{ if } d_{\mathbf{A}}\left(\mathbf{m}_j, \mathbf{c}_l^{(i)}\right) > 0, \text{ and } u_{ij}^{(l)} \in [0,1] \text{ with } \sum_{i=1}^{c} u_{ij}^{(l)} = 1$$

Until $\|\mathbf{U}^{(l)} - \mathbf{U}^{(l-1)}\| < \delta$

3. Genetic Programming for Discovering Classification Rules

GP is an extension of Genetic Algorithms (GAs). It solves the representation problem in GAs. The main principles of GP are based on the mechanisms of Evaluation.

(a) Survival of the fittest and natural selection.

(b) An offspring's chromosome consists of parts derivate from the chromosome of its parents (i.e., inheritance mechanism).

(c) A change in the offspring is a characteristic which is not inherited from the parents (i.e, mutation).

This paper proposes a genetic programming (GP) system for discovering simple classification rules in the following format: IF (a-certain-combination-of-attribute values-is-satisfied) THEN (predict-a certain-class). Each individual represents a set of these IF-THEN rules. This rule format has the advantage of being intuitively comprehensible for the user. Hence, he/she can combine the knowledge contained in the discovered rules with his/her own knowledge, in order to make intelligent decisions about the target classification problem.

The use of GP for discovering comprehensible IF-THEN classification rules is relatively little explored in the literature, by comparison with more traditional rule induction and decision-tree-induction methods [10]. We believe such a use of GP is a promising research area, since GP has the advantage of performing a global search in the space of candidate rules. In the context of classification rule discovery, in general this makes it cope better with attribute interaction than conventional, greedy rule induction and decision-tree-building algorithms [11], [12].

3.1. Individual Representation

An individual can contain multiple classification rules, subject to the restriction that all its rules have the same consequent – i.e., they predict the same class. In other words, an individual consists of a set of rule antecedents and a single rule consequent.

The rule antecedents are connected by a logical OR operator, and each rule antecedent consists of a set of conditions connected by a logical AND operator. Therefore, an individual is in disjunctive normal form (DNF) – i.e., an individual consists of a logical disjunction of rule antecedents, where each rule antecedent is a logical conjunction of conditions (attribute-value pairs). The rule consequent specifies the class to be predicted for an instance that satisfies all the conditions of any of the rule antecedents.

The terminal set consists of the attribute names and attribute values of the data set being mined. The function set consists of logical operators (AND, OR) and relational operators (“=“, “≠ “, “≤“, “>“,"-","+","*").

Figure 1 shows an example of the genetic material of an individual. Note that the rule consequent is not encoded into the genetic material of the individual. Rather, it is chosen by a deterministic procedure, as will be explained later. In the example of Figure 1 the individual contains two rules, since there is an OR node at the root of the tree. Indeed, the tree shown in that figure corresponds to the following two rule antecedents:

$$\text{IF}\,(A_1 \le 2)\ \ \text{OR}\ \ \text{IF}\,((A_3 \ne 1)\ \text{AND}\,(A_5 > 1)).$$

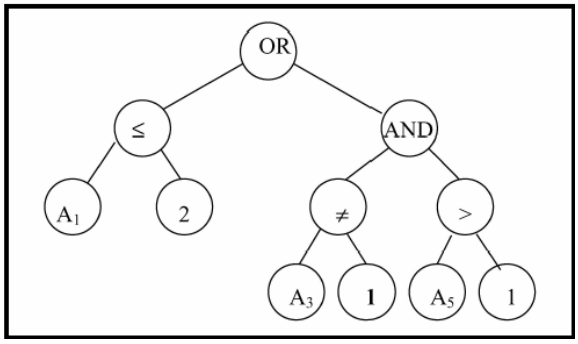


Fig 1. Example of an individual

The use of these operators in the tree associated with an individual must satisfy some constraints about the data types of these operators, as shown in Table 1.

Table 1.Data types restrictions for the function set

Operator	Input arguments	output
AND, OR	(Boolean ,Boolean)	Boolean
"=","≠"	(continuous,continuous)	Continuous
“≤“, “>“	(continuous,continuous)	Boolean
“+“, “-“, "*"	(continuous,continuous)	Continuous

Note that the relational comparison operators (“≤“, “>“) cannot be used as child nodes (i.e., these operators can be used only in the root node of a tree. To summarize, in our individual representation each individual consists of a set of rules predicting a given class and an entire solution for the classification problem consists of *k* individuals, each of them predicting a different class.

One advantage of this approach, by comparison with the previously mentioned conventional approach of running the GP once for each class, is that in the former we need to run the GP just once to discover rules predicting different classes. Therefore, our approach is considerably more efficient, in terms of computational time.

At the last we must also used a restriction on the size of the tree associated with an individual. This size restriction is important for at least two reasons. First, from a predictive data mining view point, avoiding the generation of very large trees helps to combat over fitting and so potentially improves the predictive power of the candidate attribute. Second, from a GP viewpoint, this size restriction helps to avoid the effects of code bloat [13], [14]. (i.e., the tendency of GP trees to grow in an uncontrolled manner).

3.2. Selection Method

We use tournament selection. In essence, this method works as follows. First, *k* individuals are randomly chosen from the population (i.e., that individual represent the best seed for each clusters result in fuzzy c-means clustering algorithm). Then

the individual with the best fitness is selected. This method has an important parameter, the tournament size, *k*.

This parameter determines the selective pressure of the method. Larger values of *k* correspond to larger selective pressures, favouring individuals with the best values of fitness. As will be seen later, we have done experiments with different values of this parameter, to determine how robust our GP is to variations in the setting of this parameter.

3.3. Genetic Operators

In order to create a new population from the current population we use three operators, namely reproduction, crossover and mutation. Reproduction and crossover are conventional GP operators – we use standard tree crossover.

3.3.1. The Crossover Operation

The three crossover operations:-

3.3.1.1. The Node Crossover

The following, steps of how we can implementation this method.

a) Select two parents from population.

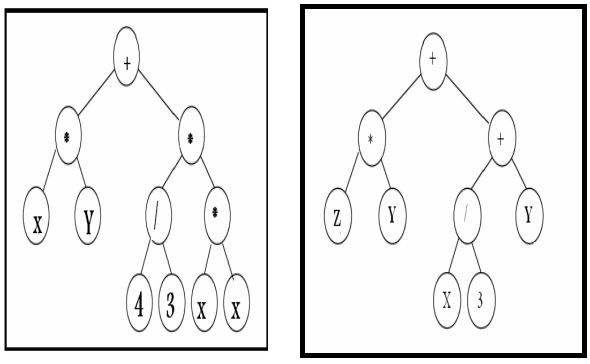
b) Select random one crossover node and the first parent and search randomly in the second parent for an exchangeable.

c) Swap the crossover node.

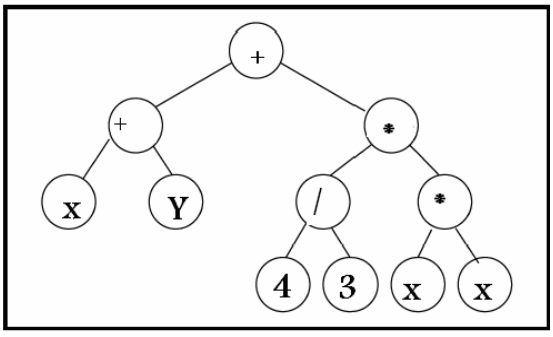
d) The child is a copy of the modified its first parent.

Example (1):- if first parent= $xy+4/3x^2$ and second parent = $zy+ x/3+y$

First parent Second parent



Child = $x+y+4/3x^2$



3.3.1.2. The Branch Crossover

The following, steps of how we can implementation this method.

a) Select two parents from population.

b) Select random one crossover node and the first parent and search randomly in the second parent for an exchangeable.

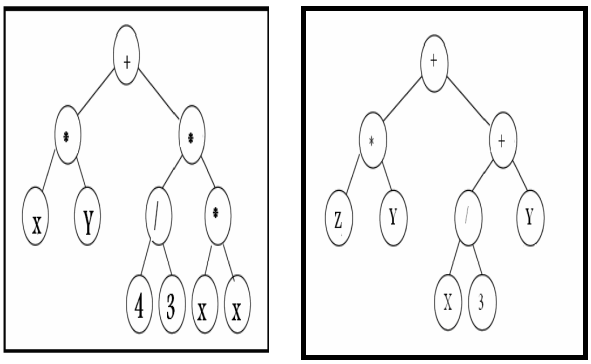
c) Cutoff the branch with the crossover nodes.

d) Calculate the size of the expected child (remind size of first parent + size of branch cutoff from the second parent).

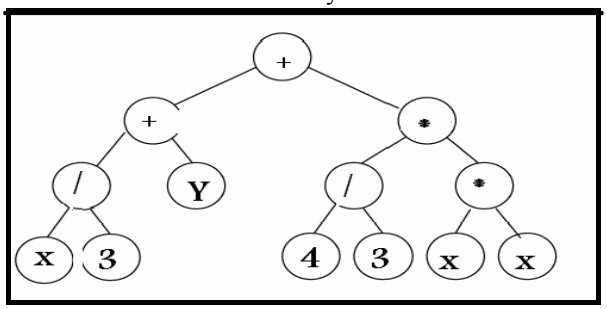
e) If the size of child is accepted created the child by appending the prang cutoff from second parent t o the ramaing of first parent otherwise try again starting from (b).

Example (2):- if first parent= $xy+4/3x^2$ and second parent = $zy+ x/3+y$

First parent Second parent



Child = $x/3+y+4/3x^2$



3.3.1.3. The Mixed Crossover

The following, steps of how we can implementation this method.

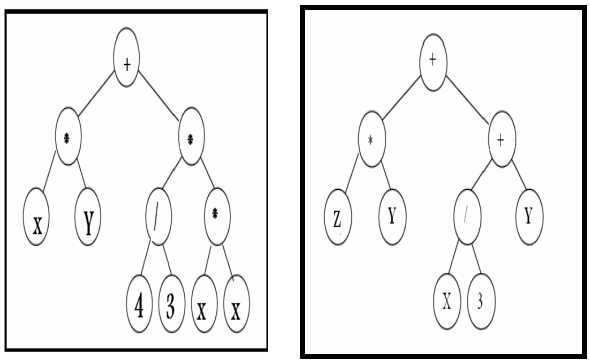
a) Select two parents from population.

b) Select random one crossover node a terminal node in the second parent.

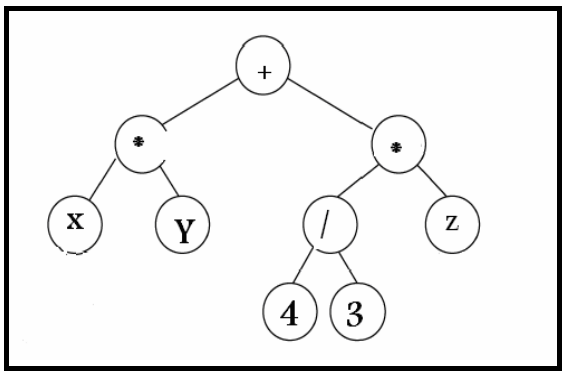
c) Generate the child by replacing the branch with the crossover node in first parent with the terminal node select from second parent.

Example (3):- if first parent= $xy+4/3x^2$ and second parent = $zy+ x/3+y$

First parent Second parent



Child = xy+4/3z



3.3.2 The Mutation Operation

The three crossover operations:-

3.3.2.1 The Node Mutation

The following, steps of how we can implementation this method.

a) Determine the mutation node.

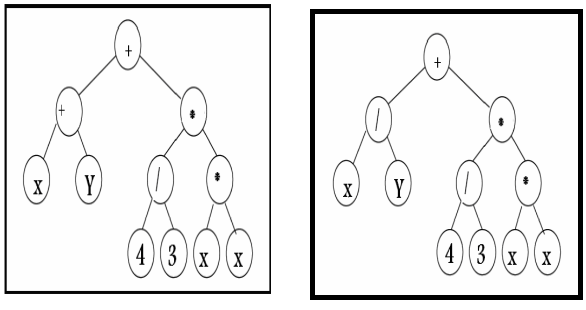
b) Select randomly an exchangeable node in the defined node vectors.

c) Swap the mutation nodes.

The mutation operator used in this article works as follows. First, it randomly chooses a tree node. Then the current symbol in this node is replaced by a randomly chosen symbol of the same kind which is different from the current symbol. More precisely, a terminal symbol is replaced by another terminal symbol, an arithmetic operator is replaced by another arithmetic operator, and a relational comparison operator is replaced by another relational comparison operator.

Example (4):-

Child= $x+y+4/3x^2$ Mutation Child=$x/y+4/3x^2$



3.3.2.2 The Shrinking Mutation

The following, steps of how we can implementation this method.

a) Determine the mutation node.

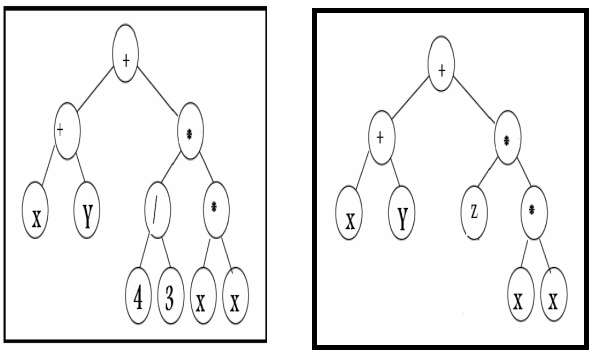
b) Select randomly terminal node in the defined node vectors.

c) Cutoff the branch with the mutation node.

d) Append the terminal node.

Example (5):-

Child= $x+y+4/3x^2$ Mutation Child=$x+y+zx^2$



3.3.2..3 The Growing Mutation

The following, steps of how we can implementation this method.

a) Determine the mutation node.

b) Cutoff the branch with the node.

c) Calculate the size of the remaining individual.

d) Generate branch as follows:-

1. Select randomly a nodes as the starting node.

2. Select randomly further nodes and append them the output of the starting node.

3. Determine the new amount of outputs according to the node protocols of appended nodes.

4. Select randomly further node and append them to the output

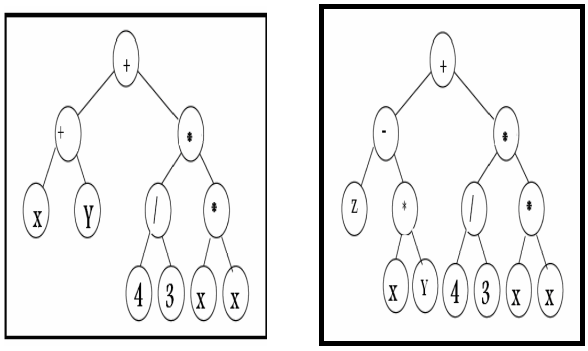
5. Repeat steps 3-4 until no free output connections are left.

e) Calculate the size of the branch.

f) If the sum of the size on the remain child and new branch is accepted, insert the new branch at the mutation nod, otherwise, return to step (a) and generated new branch.

Example (5):-

Child= $x+y+4/3x^2$ Mutation Child=$z-xy+4/3x^2$



3.4. Fitness Function

The fitness function used in this work is information gain ratio [15], which is a well-known attribute-quality measure in the data mining and machine learning literature. It should be noted that the use of this measure constitutes a data-driven strategy. As mentioned above, an important advantage of this kind of strategy is that it is relatively fast, since it avoids the need for running a data mining algorithm when evaluating an attribute (individual). In particular, the information gain ratio for a given attribute can be computed in a single scan of the training set.

The Information Gain Ratio of an attribute *A*, denoted by IGR(*A*), is computed by dividing the Information Gain of *A*, denoted by IG(*A*), by the amount of Information of the attribute *A*, denoted I(*A*), i.e.:

$$IGR(A) = IG(A) / I(A) \quad (5)$$

The Information Gain of an attribute *A*, denoted IG(*A*), represents the difference between the amount of Information of the goal (class) attribute *G*, denoted I(*G*), and that amount given the knowledge of the values of an attribute *A*, denoted I(*G*|*A*). IG(A) is given by:

$$IG(A) = I(G) - I(G/A) \quad (6)$$

Where

$$I(G) = -\sum_{j=1}^{n} p(G_j) \cdot \log_2 p(G_j) \quad (7)$$

And

$$I(G|A) = \sum_{i=1}^{m} p(A_i) \left(-\sum_{j=1}^{n} p(G_j|A_i) \cdot \log_2 p(G_j|A_i) \right) \quad (8)$$

Where p(*Gj*) is the estimated probability (computed in the training set) of observing the *j*-th class (i.e., the *j*-th value of the goal attribute *G*), *n* is the number of classes,

p(*Ai*) is the estimated probability of observing the *i*-th value of the attribute *A*, *m* is the number of values of the attribute *A*, and p(*Gj*|*Ai*) is the empirical probability of observing the *j*-th class conditional on having observed the *i*-th value of the attribute *A*.

Finally, I(*A*) is given by:

$$I(A) = -\sum_{i=1}^{m} p(A_i) \cdot \log_2 p(A_i) \quad (9)$$

For more details about this procedure and the information gain ratio measure in general, see [15].

4. Computation Result

In this section we report the results of computational experiments performed to evaluate our proposed method for knowledge discovering in database. The experiments were performed with five public-domain data sets from the UCI (University of California at Irvine) data set repository.

Table 2 shows the main characteristics of the data sets used in our experiments.

Table 2.Dataset information

Data set	No. of attributes	No. of classes	No. of records
Breast Cancer	10	2	699
Segmentation	19	7	210
Glass	10	2	214
Zoo	18	7	101
Iris	4	3	150

In all our experiments we used the following parameters of fuzzy c-means clustering algorithm :- w=[1.2,1.25,2.5,1,4.5],δ=[0.01,0.0009,0.0007,0.00036 ,0.001],d=[699,210,214,101,150],I=[20,35,12,21,50].

The probabilities of crossover and mutation were 90% and 10%, respectively.

In addition to the goal of evaluating the quality of the knowledge discovering in data mining by the FCM & GP, our experiments also had the goal of determining how robust the GP is to variations in the setting of two important parameters, namely the tournament size *k* and the maximum tree size (number of nodes). Hence, we did experiments with three different values for the tournament size k (2, 4, 8) and three different values for the maximum tree size (18, 32, 64). The experiments involved all the 9 possible combinations (3 x 3) of values of these two parameters.

Some of the best results of the final population (after 50 generations) of the genetic programming that produced the resules (knowledge discovering in data mining) are explaining below, where each class represented by if-then rules .

1. Cancer Beast

IF (Clump Thickness = 8) AND (Uniformity of Cell Size = 7) AND (Uniformity of Cell Shape = 7) AND (Marginal Adhesion = 5) AND (Single Epithelial Cell Size = 5) AND (Bare Nuclei = 10) AND (Bland Chromatin = 7) AND

(Normal Nucleoli = 6) AND (Mitoses= 1) THEN Class is MALIGNANT

IF (Clump Thickness = 5) AND (Uniformity of Cell Size = 8) AND (Uniformity of Cell Shape = 7) AND (Marginal Adhesion = 7) AND (Single Epithelial Cell Size = 6) AND (Bare Nuclei = 8) AND (Bland Chromatin = 7) AND (Normal Nucleoli = 7) AND (Mitoses = 7) THEN Class is MALIGNANT

IF (Uniformity of Cell Size = 1.38095) AND (Uniformity of Cell Shape =1.46939) AND (Marginal Adhesion = 1.38776) AND (Single Epithelial Cell Size = 2.13605) AND (Bare Nuclei = 1.40816) AND (Bland Chromatin =2.10544) AND (Normal Nucleoli = 1.31633) AND (Mitoses = 1.07823) THEN Class is BENIGN

IF (Clump Thickness = 3) AND (Uniformity of Cell Size= 1) AND (Uniformity of Cell Shape= 1) AND (Marginal Adhesion = 1) AND (Single Epithelial Cell Size = 2) AND (Bare Nuclei = 1) AND (Bland Chromatin =2) AND (Normal Nucleoli = 1) AND (Mitoses = 1) THEN Class is BENIGN

IF (Clump Thickness = 7) AND (Uniformity of Cell Size = 7) AND (Uniformity of Cell Shape = 7) AND (Marginal Adhesion = 7) AND (Single Epithelial Cell Size = 2) AND (Bare Nuclei = 2) AND (Bland Chromatin =2) AND (Normal Nucleoli = 2) AND (Mitoses = 2) THEN Class is BENIGN

2. Segmentation

IF (Region-Centroid-Col between [11.0-296]) AND (Region-Centroid-Row > 55.0) AND (Region-Pixel-Count = 9) AND (Short-Line-Density-5 ≠1.0) AND (Short-Line-Density-2 =0.0) AND (Vedge-Mean ≤ 1.3333335) AND (Vegde-Sd = 0.80000025) Hedge-Mean = 0.95185167) AND (Hedge-Sd > 17.666666) AND (Intensity-Mean = 19.0) AND (Rawred-Mean = 21.11111) AND (Rawblue-Mean= 12.888889) AND (Rawgreen Mean > 12.888889) AND (Exred-Mean= 4.0) AND (Exblue-Mean >10.33333) AND (Exgreen-Mean ≤ -14.333333) AND (Value-Mean= 21.11111) AND (Saturatoin-Mean = 0.38875645) AND (Hue-Mean = 1.3021333) THEN Class is BRICKFACE.

IF (region-centroid-col = 228.0) AND (region-centroid-row = 20.0) AND (region-pixel-count = 9) AND(short-line-density-5 = 0.0) AND (short-line-density-2 = 0.0) AND (vedge-mean = 1.0555547) AND (vegde-sd = 0.49065518) AND (hedge-mean = 0.8333333) AND (hedge-sd > 0.2277338) AND (intensity-mean = 125.0) AND rawred-mean = 114.0) AND (rawblue-mean = 140.55556) AND (rawgreen-mean = 120.44444) AND (exred-mean ≤ -33.0) AND (exblue-mean = 46.666668) AND (exgreen-mean = -13.666667) AND (value-mean = 140.55556) AND (saturatoin-mean = 0.18889166) AND (hue-mean = 2.348006) THEN Class is SKY.

IF (region-centroid-col = 9.0) AND (region-centroid-row = 80.0) AND (region-pixel-count = 9) AND (short-line-density-5 = 0.0) AND (short-line-density-2 = 0.0) AND (vedge-mean =2.944444) AND (vegde-sd =13.751853) AND(hedge-mean = 16.666666) AND(hedge-sd = 71.5111) AND (intensity-mean = 23.62963) AND (rawred-mean = 17.333334) AND (rawblue-mean =31.666666) AND (rawgreen-mean=21.88889) AND (exred-mean =-18.88889) AND (exblue-mean = 24.11111) AND (exgreen-mean =-5.2222223) AND (value-mean = 31.666666) AND (saturatoin-mean = 0.5142537) AND (hue-mean=-2.4315135) THEN Class is FOLIAGE.

IF (region-centroid-col = 79.0) AND (region-centroid-row=28.0) AND (region-pixel-count = 9) AND (short-line-density-5=0.0) AND (short-line-density-2 =0.0) AND (vedge-mean = 4.277777) AND(vegde-sd = 3.7618961) AND (hedge-mean = 0.8333333) AND (hedge-sd = 0.6582806) AND (intensity-mean = 62.407406) AND(rawred-mean = 53.444443) AND (rawblue-mean = 79.111115) AND (rawgreen-mean = 54.666668) AND (exred-mean = 26.88889) AND (exblue-mean =50.11111) AND(exgreen-mean = -23.222221) AND (value-mean = 79.111115) AND(saturatoin-mean = 0.32455578) AND (hue-mean = -2.1445074) THEN Class is CEMENT.

IF (region-centroid-col=189.0)AND(region-centroid-row=141.0) AND (region-pixel-count=9) AND (short-line-density-5= 0.0) AND (short-line-density-2=0.0) AND (vedge-mean=0.0) AND (vegde-sd=0.0) AND (hedge-mean=0.0) AND (hedge-sd=0.0) AND (intensity-mean=0.0) AND (rawred-mean=0.0) AND (rawblue-mean=0.0) AND (rawgreen-mean=0.0) AND (exred-mean=0.0) AND (exblue-mean=0.0) AND (exgreen-mean=0.0) AND (value-mean=0.0)AND (saturatoin-mean=0.0) AND (hue-mean=0.0) THEN Class is WINDOW

IF (region-centroid-col=198.0) AND (region-centroid-row=183.0) AND (region-pixel-count= 9) AND (short-line-density-5=0.0) AND (short-line-density-2=0.0) AND (vedge-mean=1.0555547) AND (vegde-sd=1.1816497) AND (hedge-mean=3.3888881) AND (hedge-sd=1.5974506) AND (intensity-mean=54.037037) AND (rawred-mean=48.88889) AND (rawblue-mean=66.55556) AND (rawgreen-mean= 46.666668) AND (exred-mean= -15.444445) AND (exblue-mean =37.555557) AND (exgreen-mean= -22.11111) AND (value-mean= 66.55556) AND (saturatoin-mean=0.2986217) AND (hue-mean= -1.9749589) THEN Class is PATH.

IF (region-centroid-col= 186.0) AND (region-centroid-row=218.0) AND (region-pixel-count=9) AND (short-line-density-5= 0.0) AND (short-line-density-2=0.0) AND (vedge-mean=1.1666666) AND (vegde-sd =0.74444425) AND (hedge-mean=1.1666665) AND (hedge-sd= 0.65555507) AND (intensity-mean=13.703704) AND (rawred-mean=10.666667) AND (rawblue-mean=12.666667) AND (rawgreen-mean=17.777779)AND (exred-mean=-9.111111) AND (exblue-mean=,-3.1111112) AND (exgreen-mean = 12.222222) AND (value-mean=17.777779) AND (saturatoin-mean = 0.40134683) AND (hue-mean=2.3826835) THEN Class is GRASS .

3. Iris

IF (sepal length >4.3 OR sepal length ≤5.8) AND (sepal width ≤ 4.4) AND (petal length =1.9) AND (petal width >0.1) THEN Class is IRIS SETOSA.

IF (sepal length =6.7) AND (sepal width≤ 2.5) AND (petal length>3.0) AND (petal width=1.7) THEN Class is IRIS VERSICOLOUR.

If (sepal length > 4.9 OR sepal length ≤ 4.9) AND (sepal width =3.8 OR sepal width between [3.5-2.2]) AND (petal length >4.5 AND petal length ≤6.9) AND petal width between [2.5-1.4]) THEN Class is IRIS VIRGINICA.

5. Conclusion

This paper shows that the use of fuzzy c –means clustering technique prove fast and accrue clustering process. Also it shows that GP is more suitable tool to construct rules and verify them. We so that the extracted knowledge is clearly understandable and have a good generality.

Future work will apply our approach with injection of a kwon data mining methods.

REFERENCES

1. Hu, Y-J. *A Genetic Programming Approach to Constructive Induction* . In Proceeding of 3rd Anual Genetic Programming Conference, pp. 146–151, 1998.

2. Pagallo, G. & Haussler, D. *Boolean Feature Discovery in Empirical Learning*. In Machine Learning 5, pp. 71–99. 1990.

3. Zheng, Z. *Constructing X-of-N attributes for decision tree learning*. Machine Learning 40 (2000), 1-43.

4. Padhraic Smyth Usama M. Fayyad, Gregory Piatetsky-Shapiro. *From data mining to knowledge discovery: An overview*. In Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press, 1996.

5. U. Fayyad and R. Uthurusamy, *Data mining and knowledge discovery in databases*, Commun. ACM, vol. 39, pp. 24–27, 1996.

6. S. Mitra, P. Mitra, and S. K. Pal, *Data Mining In Soft Computing Framework: A Survey*, IEEE Transactions On Neural Networks, Vol.13, No. 1, January 2002.

7. Zadeh.L, 1965. Fuzzy Set "Information and control, 8, pp.338-352.

8. Bezdek.J, 1981.*Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New Youk.

9. Ruspini.E, 1969.*A New Approach to Clustering, Information and Control*, 15, pp.22-32.

10. Witten. I. and Frank. E. Data Mining: practical machine learning tools and techniques with Java implementations. San Mateo: Morgan Kaufmann, 2000.

11. Freitas.A. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer, 2002.

12. Papagelis.A and Kalles.D. *Breeding decision trees using evolutionary techniques*. *Proc. 18th Int. Conf. on Machine Learning*, 393-400. San Mateo: Morgan Kaufmann, 2001.

13. Langdon. W. *Quadratic bloat in genetic programming*. Proc. 2000 Genetic and Evolutionary Computation Conf. (GECCO-2000), 451-458. Morgan Kaufmann, 2000.

14. Langdon, W.B.; Soule, T.; Poli, R. and Foster, J.A. *The evolution of size and shape*. In: L. Spector, W.B. Langdon, U-M. O'Reilly and P.J. Angeline. (Eds.) *Advances in Genetic Programming* Volume 3, 163-190. MIT Press, 1999.

15. Quinlan, J.R. C4.5: *Programs for Machine Learning*. Morgan Kaufmann, 1993.