# A Framework for Optimizing Extended Belief Rule Base Systems with Improved Ball Trees

Yang-Geng Fu[a], Jin-Hui Zhuang[a], Yu-Peng Chen[a], Long-Kun Guo[a,b], Ying-Ming Wang[c],

[a] *College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China*

[b] *Shandong Key Laboratory of Computer Networks, School of Computer Science and Technology, Shandong Computer Science Center, Qilu University of Technology (Shandong Academy of Sciences), Jinan, 250353, P.R. China*

[c] *Institute of Decision Sciences, Fuzhou University, Fuzhou 350116, China*

**Abstract**

Decision support systems imposed a challenge of efficiently reasoning through utilizing stored unordered rules. The Extended Belief Rule-Base(EBRB) system, renowned for its capability of modeling data with vagueness, incompleteness, uncertainty and nonlinear features, requires to traverse over all the rules for the execution of reasoning and hence suffers reduced inference accuracy and efficiency. To improve the performance, we propose a framework of storing and retrieving belief rules based on employing an optimized structure called improved Ball tree. The framework first constructs a Ball tree index according to the distance between different rules in the metric space via using the $k$-means++ algorithm, whereas traditional Ball tree employs the $k$-centers algorithm. Then, through our proposed dynamic threshold radius adjusting method, our algorithm finds an appropriate dataset threshold and consequently activates more related rules. In the mean time, the number of retrieved and activated irrelevant rules is significantly reduced, resulting in consequently improved reasoning accuracy and efficiency. Lastly, three sets of experiments were carried out to validate our algorithm in comparison with EBRB.

*Keywords:* Ball tree, Extended Belief Rule-Base, Evidential Reasoning

## 1. Introduction

The extended belief rule base (EBRB) system proposed by Liu et al.[12] is attracting considerable research interest in the community of decision support systems because of its advantage in reasoning over data with vagueness, incompleteness, uncertainty and nonlinearity. In general, an EBRB system consists of two key components: the extended belief rule base and the evidential reasoning (ER) algorithm[25][22]. The former is a knowledge base which stores various types of belief rules, possibly with vague, uncertain and incomplete information. In essential, the base is obtained by adding a belief structure to the traditional method base of the IF-THEN rule [20]. The latter component retrieves the stored rules and activates the key rules, and then synthesizes the retrieved rules according to Dempster's criteria to obtain the system's reasoning results[6].

A fundamental challenge in EBRB is how to select a desirable set of key rules to activate. Traditionally, the similarity of the input data with individual rule in EBRB determines which rules should be activated. Naive methods for the task, aiming to eventually find the set of rules that are most similar to the input data, require a traverse over all the rules in the entire rule base. Apparently, such methods are inefficient and consequently bring a problem — how to store and index the rules, such that we could retrieve the most similar rules for the input data in a more efficient way. Clearly, storing and indexing the rules in alphabeta order is not an option because it will not accelerate the process as the retrieval is based on similarity.

For the task of storing and indexing rules, many tree structures have been adopted by researchers for its simplicity and efficiency, including BK tree [19], KD tree [29] , VP tree, MVP tree[11], etc. However, the embedded algorithms for the computing the threshold requires a large number of iterations, and hence suffer compromised efficiency. Comparing with the above tree structures, Ball tree can achieve better indexing efficiency particularly when against queries based on similarity and hence has been intensively studied in literature [13]. For example, Ball tree has been employed by applications such as $k$-anonymity [5] and image matching [21]. However, to the best of our knowledge, Ball tree has not yet been used in EBRB system. That is mainly because there exist few methods for efficiently calculating the threshold radius for tree structures in EBRB system, which is critical for the quality of a constructed Ball tree. Therefore, the main purpose of the paper is to devise an efficient algorithm for adaptively calculating the threshold, and improve Ball tree so that it better suits the indexing and retrieval within EBRB system.

### *1.1. Main results*

In the paper, we will propose a scheme for construction and retrieval of rule base by employing improved Ball trees and a novel method for computing the threshold radius. The results can be summarized as in the following:

- Improve the Ball tree indexing method by firstly using $k$-means++ to divide the rules into clusters instead of the $k$-centers method as in the traditional Ball tree method, and secondly employing simulated annealing to calculate the proper radius of the subball, such that the computed balls have a notably reduced radius.

- Devise a novel retrieval method to find the $k$ nearest rules for any given rule. Our method combines the exponential increasing method with the $k$-nearest neighbor retrieval method. Comparing to the previous method with KD-tree and BK-tree as building blocks, our retrieval and reasoning execution based on the constructed Ball tree has a significant reduced runtime and a comparable accuracy rate.

- Extensive numerical experiments were carried out to demonstrate the practical performance gain of our method against both simulation and real-world applications.

### *1.2. Organization*

The organization of this paper is as follows. Section 2 surveys the existing related literature; Section 3 introduces the two key technologies involved in this paper: first the establishment and reasoning process of EBRB system, and then the data structure of the traditional Ball tree algorithm; Section 4 presents the implementation principle of our proposed BTDCS-EBRB system based on the improved Ball tree algorithm. Section 5 evaluates our proposed BTDCS-EBRB system via comparing with previous state-of-art algorithms as baselines through running three sets of numerical experiments; And lastly, Section 6 concludes this paper.

## 2. Related work

EBRB was first proposed by Liu et al[12] for solving the challenges of combination explosion and complex parameter training existing in the famous BRB system[23][3][10][33]. By employing D-S evidence theory[6], fuzzy theory[31][8], and decision theory[31], EBRB is shown deserves satisfactory accuracy in oil pipeline leak detection for its first application [12]. Later, besides the further improved applications in oil pipeline leak detection as in [27], EBRB was also used for health estimation[2], environmental governance cost prediction[23], single sensor architecture configuration optimization[7], classification problems[26] and so on.

Despite the broad applications, one of the important challenge in EBRB system is how to achieve better rule retrieval effectiveness and efficiency, as well as the consequent reasoning speed and accuracy. Starting from the perspective of optimizing the system structure, Calzada et al.[1] proposed a dynamic rule activation method (DRA). Through iteratively adjusting parameters for activating rules, the methods successfully achieve the purpose of activating more relevant rules. However, the parameter iterations result in significant time and space consumption. Yu et al.[30] introduced the 80/20 principle, and accordingly used the rules with the top 20% activated weights instead of all the activated rules. Yang et al.[28] proposed a data envelopment method to reduce the rules within the initial rule base, and then used the remaining rules to execute the reasoning process. However, both of these reducing processes inevitably lead to loss of data information.

Therefore, the structural optimization of the rule base has become a hot topic of discussion in EBRB system optimization. Because of its simple and efficient characteristics, the tree structures are intensively employed for optimizing the structure of the EBRB system so as to improve the inferencing speed and accuracy of the system. Su et al.[19] proposed a Structure Optimization Framework of Extended Belief Rule Base based on BK-Tree (hereinafter referred to as the BK-EBRB system). The BK-EBRB system retrieves rules that are similar to the input data according to a searching threshold, and activates these neighbor rules for final ER synthesis reasoning, eventually achieving better speed of EBRB's rule retrieval and higher reasoning accuracy. However, it remains open to design methods for determining the appropriate searching threshold. As a result, the performance of BK-EBRB has no guarantee when the computed threshold is not as good. By integration of BK trees and KD trees, Yang et al.[29] developed a Multi-attribute

Search Framework, which flexibly switches between the two tree structures mainly according to the dimensionality of the data, achieving higher accuracy and fast retrieval efficiency as manifested through experiments. However, the method has poor search efficiency on high-dimensional data, and would possibly fails adapting different data characteristics. The EBRB system based on the VP tree and MVP tree proposed by Lin et al.[11] uses the k-means method to determine the selection threshold. However, the appropriate number of clusters is hard to determine when against different circumstances. In addition, the representative rule of each cluster is randomly selected, which can not guarantee the accuracy of the system's reasoning.

In this paper, we will use Ball tree as the structure for indexing and retrieving rules. Ball tree[15] is a multidimensional binary search tree which uses hypersphere partition instead of hyperrectangular partition. Consequently, Ball tree brings better partitioning of (especially high-dimensional) data, although requires higher space complexity for its construction. Ball tree has already been use for many applications that uses similarity as discrimination metrics [16][17]. More recently, Cheng et al.[5] used the structure of Ball tree to enhance the $k$-anonymity algorithm in the privacy data protection model, so as to boost identifying the sets of $k$ similar points. Wan et al.[21] used Ball tree to index image for retrieval, and significantly improved the efficiency of image matching.

## 3. Preliminary and Notations

In this section, we will first introduce the structure of EBRB with the reasoning procession therein, and then address Ball tree as well as its traditional constructing framework.

### *3.1. Representation of EBRB*

EBRB is based on BRB, in order to better integrate the fuzzy, uncertain and contradictory information in the data into the belief rules. In addition to embedding belief degrees in the referential value of each antecedent attribute of the extended belief rule, the extended belief rule base also embeds belief degrees in the consequent referential value of each rule.

In general, the expression of the $k^{th}$ rule in the extended belief rule base is as follows:

$$IF\, A_1^k \wedge A_2^k \wedge ... \wedge A_{T_k}^k\, THEN\, \{(D_1, \beta_1^k), (D_2, \beta_2^k), \ldots, (D_N, \beta_N^k)\} \tag{1}$$

where $A_i^k(i = 1, 2, \dots, T_k, k = 1, 2, \dots, L)$ is the referential value set of the $i^{th}$ antecedent attribute in the $k^{th}$ rule, $D_n(n = 1, 2, \dots, N)$ is the referential value of the $n^{th}$ consequent attribute, $T_k$ is the number of antecedent attributes of the $k^{th}$ rule, and $\beta_n^k(0 \leq \beta_n^k \leq 1)$ represents the belief degree of the referential value $D_n$ in the $k^{th}$ rule. The antecedent attribute $A_i^k$ can be further formulated as below:

$$\{\{(A_{i,j}^k, \alpha_{i,j}^k), j = 1, 2, \dots, J_i\}|i = 1, 2, ..., T\}(k = 1, 2, ..., L) \tag{2}$$

where $\alpha_{i,j}^k$ indicates the belief degree of the $j^{th}$ referential value corresponding to the $i^{th}$ antecedent attribute in the $k^{th}$ rule, $J_i$ denotes the number of the $i^{th}$ antecedent attribute referential value and $L$ is the number of rules.

Similar to BRB, the integrity of the rule in EBRB can be established by $\sum_{j=1}^{N} \beta_j^k$. If $\sum_{j=1}^{N} \beta_j^k = 1$, then the $k^{th}$ rule is complete, otherwise, the $k^{th}$ rule is incomplete. The difference between EBRB and BRB is that the EBRB has two additional parameters $\theta_k$ and $\delta_i$, where $\theta_k$ and $\delta_i$ represent the rule weight of the $k^{th}$ rule and the weight of the $i^{th}$ antecedent attribute of the rule, respectively. The former weight reflects the importance of the rule among all extended belief rules, and the latter reflects the importance of the $i^{th}$ antecedent attribute in all the antecedent attributes of this rule.

### *3.2. Establishment of EBRB*

For building EBRB, Liu et al. [12] pioneered a data-driven approach whose main steps proceed as follows:

Step 1: Determine the set of referential values of the antecedents attributes of the EBRB system by employing the utility-based calculation method of the belief structure. The set can be given by an expert or determined by a fuzzy membership function, where the number of referential values $A_{i,j}$ is denoted by $\gamma i, j\ means\ |A_{i,j}|, j = 1, 2, \dots, J_i$. Then the input value can be formulated as the following mathematical expectations:

$$S(x_i) = \{(\gamma_{i,j}, \alpha_{i,j}), i = 1, 2, \dots, T_k, j = 1, 2, \dots, J_i\} \tag{3}$$

Step 2: Establish a belief structure between the input data and the referential value. According to the given input data, the belief degree of the antecedent attribute and the consequent attribute of the extended belief rule can be calculated according to the following formula:

$$\alpha_{i,j} = \frac{\gamma_{i,j+1} - x_i}{\gamma_{i,j+1} - \gamma_{i,j}}, \gamma_{i,j} \leq x_i \leq \gamma_{i,j+1}, j = 1, 2, \dots, J_i - 1 \tag{4}$$

$$\alpha_{i,j+1} = 1 - \alpha_{i,j} \tag{5}$$

Step 3: Determine the weight of the antecedent attributes and the weight of each extended belief rule. Note that inconsistencies of the rules would exist during the calculation of rule weights, as we generate the rules from the data with various inconsistencies. So we adjust the weight of the rules by using the method proposed by Liu et al.[12].

*3.3. Reasoning of EBRB system*

During the execution of EBRB reasoning, the input data is first formulated in the form of belief structure. Then the result of the inference is mainly obtained in two steps: the calculation of the activated weight and the synthesis of the activated rule.

When calculating the activated weight of each rule, we enumerate all the rules that have been constructed, and calculate the distance between the data and the antecedent attribute belief structure matrix of each rule by the activated weight calculation Equations (6-8) as in the following:

$$d_i^k = \sqrt{\sum_{j=1}^{J_i} (\alpha_{i,j} - \alpha_{i,j}^k)^2} \tag{6}$$

$$S_i^k = 1 - d_i^k \tag{7}$$

$$\omega_k = \frac{\theta_k \prod_{i=1}^{T_k} (S_i^k)^{\bar{\delta}_i}}{\sum_{l=1}^{L} [\theta_l \prod_{i=1}^{T_k} (S_i^l)^{\bar{\delta}_i}]}, \bar{\delta}_i = \frac{\delta_i}{max_{i=1,2,\ldots,T_k}\{\delta_i\}} \tag{8}$$

where $0 \leq \omega_k \leq 1$, $\sum_{i=1}^{L} \omega_i = 1$. We use $\omega_k = 0$ to denote the case the $k^{th}$ rule is not activated; and $0 < \omega_k \leq 1$ otherwise. Then the basic trusted value of the evaluation result can be calculated according to Equation(9-12) for the activated rule.

$$m_{n,i} = \omega_i \beta_{n,i} \tag{9}$$

$$m_{H,i} = 1 - \omega_i \sum_{n=1}^{N} \beta_{n,i} = \bar{m}_{H,i} + \widetilde{m}_{H,i} \tag{10}$$

$$\widetilde{m}_{H,i} = \omega_i(1 - \sum_{n=1}^{N} \beta_{n,i}) \tag{11}$$

$$\bar{m}_{H,i} = 1 - \omega_i \tag{12}$$

For the phase of synthesizing the activated rules, we use the ER synthesis Equation (13-18) as in the following:

$$C_n = k[\prod_{j=1}^{L}(m_{n,j} + \bar{m}_{H,j} + \widetilde{m}_{H,j}) - \prod_{j=1}^{L}(\bar{m}_{H,j} + \widetilde{m}_{H,j})] \tag{13}$$

$$\widetilde{C}_H = k[\prod_{j=1}^{L}(\bar{m}_{H,j} + \widetilde{m}_{H,j}) - \prod_{j=1}^{L}\bar{m}_{H,j}] \tag{14}$$

$$\bar{C}_H = k\prod_{j=1}^{L}\bar{m}_{H,j} \tag{15}$$

$$k^{-1} = \sum_{n=1}^{N}\prod_{j=1}^{L}(m_{n,j} + \bar{m}_{H,j} + \widetilde{m}_{H,j}) \qquad - (N-1)\prod_{j=1}^{L}(\bar{m}_{H,j} + \widetilde{m}_{H,j}) \tag{16}$$

$$\beta_n = \frac{C_n}{1 - \bar{C}_H}(n = 1, ..., N) \tag{17}$$

$$\beta_H = \frac{\widetilde{C}_H}{1 - \bar{C}_H}(n = 1, ..., N) \tag{18}$$

Finally, we can obtain the inference of EBRB with belief structure according to Equation 19 as in the following:

$$S(x) = (D_j, \beta_j), j = 1, 2, ..., N \tag{19}$$

## *3.4. Ball tree algorithm*

A Ball tree is a binary tree in which each node is a D-dimensional hypersphere. Designating for finding K-Nearest Neighbor (KNN), the Ball tree algorithm recursively divides the data into nodes defined by centroid $C$ and radius $r$ such that each point in the node is located within the hypersphere defined by $r$ and $C$. Note that hyperspheres of different nodes may intersect. So in order to reduce the number of candidate points for neighbor retrieve, we implement the Ball tree algorithm with respect to the triangle inequality.

In this paper, we employ the Ball tree construction method proposed by Andrew Moore et al.[14]. For briefness, we assume that the elements are with two dimensions, say $\{(x_i, y_i), i = 1, 2, ..., n\}$, and measured by Euclidean distance. Then the approach simply proceeds as follows:

Step 1: For the current point set, calculate the mass center $P(x_{mid}, y_{mid})$ of the point set by Equation(20-21)

$$x_{mid} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{20}$$

$$y_{mid} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{21}$$

Step 2: Find the minimum radius, such that all points are covered by the circle entering at point $P$.

Step 3: Randomly pick a point $P_{rd}$ in each point set, and Calculate the point $A(x_a, y_a)$ farthest from $P_{rd}$.

Step 4: Find the farthest point $B(x_b, y_b)$ from point $A$.

Step 5: Divide the current point set divided into $SetA$ and $SetB$ according to a point is closer to point $A$ or point $B$, and use $SetA$ and $SetB$ as the left and right children nodes of the current node, respectively.

Step 6: Repeat the above steps recursively until the number of points therein is less than the threshold $M$.

After building the Ball tree, it remains to give a retrieval approach. Firstly, we will retrieve the first $k$ points that are close to point $P(x, y)$ which radius is $r$. Let $O$ be the circle with point $P$ as its center and $r$ as radius.

Step 1: Compute circles accommodating the left and right children nodes; located has an intersection with the circle $O$.

Step 2: If there is an intersection between the left children node and the circle $O$, then retrieve the left children node; Otherwise, retrieve the right children.

Step 3: If the leaf nodes are retrieved, compare all the points in the leaf node point set with point $P$. If the distance in a point set is smaller than the $k^{th}$ point already found, then the $k^{th}$ point is replaced by the newly found point.

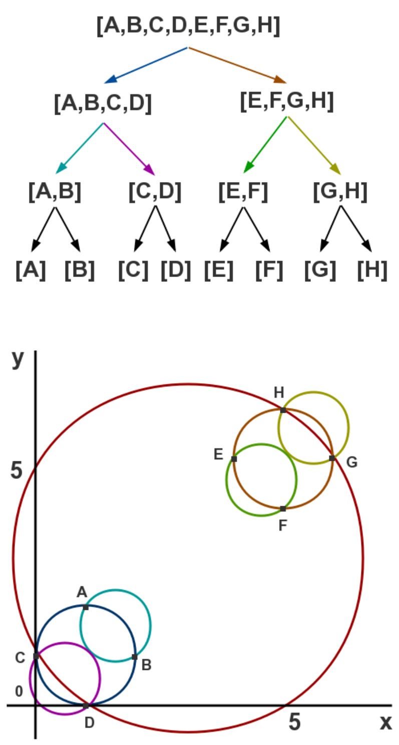

Figure 1: The establishment of the Ball tree

By repeating the above steps recursively, we can find the $k$ points closest to point $P$. Let's take a example:

Suppose there is a two-dimensional point set, say $\{A(1,2), B(2,1), C(0,1), D(1,0), E(4,5), F(5,4), G(6,5), H(5,6)\}$. Assume the point set threshold $M$ is 1. The algorithm proceeds as below:

Firstly, calculate the mass center $Mid = (2.625, 2.5)$ of the point set, and use $Mid$ to find the radius of the smallest circle $O$ covering all points.

Secondly, randomly pick a point in the point set, say point $A$, and find the point farthest from point $A$ (point $G$ in the example), and then find the point which farthest from point $G$ (point $C$ in the case).

After finding the two desired points $C$ and $G$, divide the points into two subsets according to their distances to $C$ and $G$.

Lastly, repeat the division until there remain at most $M$ points. The index structure established by the Ball tree is as demonstrated in Figure 1.

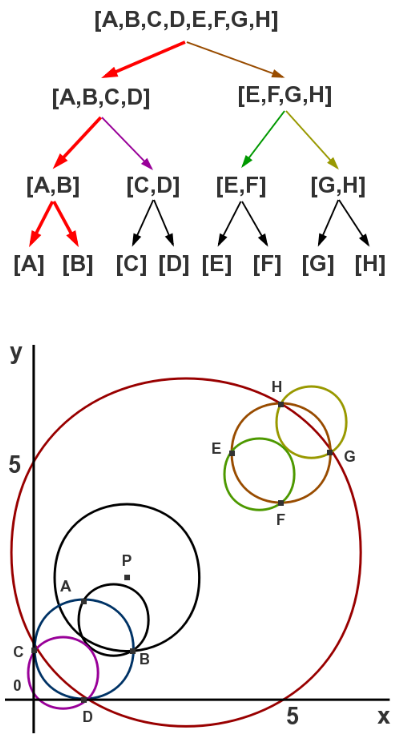

Figure 2: Retrieval of the Ball tree

## 4. Optimization of EBRB system via Ball tree index

EBRB expert system needs to traverse all the out-of-order rules in the rule base for making inferences. This results in seriously reduced efficiency of EBRB particularly when there are many rules. In this section, we propose an extended belief rule via optimizing the index and retrieval of rules based on the improved Ball tree algorithm. That is, we first index disordered rules through tree structure according to the metric distance between the rules, then through setting an appropriate threshold radius for retrieving belief rules, we activate more relevant belief rules and consequently improve the accuracy and efficiency of the EBRB system inference. For determining the appropriate threshold radius for different data sets, this paper proposes an algorithm with adaptive retrieval radius to achieve the generalization ability of the Ball tree, hereinafter referred to as BTDCS-EBRB.

**Algorithm 1:** Construction of BTDCS-EBRB

```
Input: $ruleSet, maximum\ number\ of\ rules\ that\ the\ leaf\ node$
Output: $ballTree$
Function GetRuleSetPivot($ruleSet$)
Function SplitRuleSet($ruleSet$)
if $ruleSet.size \leq maxNum$ then
    $root$ := create $leafnode$ with rules in $ruleSet$
    return $root$
else
    $root$ := create tree node
    $root.pivot$ := GetRuleSetPivot($ruleSet$)
    $root.bestRadius$ := max(distance between $ruleSet$ and $pivot$)
    $subSet[]$ := SplitRuleSet($ruleSet$)
    $root.leftChild$ = Build($subSet[1]$)
    $root.rightChild$ = Build($subSet[2]$)
    return $root$
end
```

## *4.1. Construction of BTDCS-EBRB*

Because BTDCS-EBRB system only indexes the generated rules through the Ball tree, the steps through the rule generation are the same as that in traditional EBRB System.

The Euclidean distance is usually chosen as the distance of the metric when building the Ball tree. Since the belief rule is a distributed representation, if the rule has $i$ antecedent attributes and each antecedent attribute has $j$ reference values, then the representation of an extended belief rule has $i \times j$ dimensions. Note that the weight of many premise attribute reference values is 0, which will waste the system storage space and computing time. Therefore, this paper uses the method due to Yang et al. [29] based on formula(22) for converting attribute values.

$$y_i^k = \sum_{j=1}^{J_i}(\alpha_{i,j}^k \bar{\alpha}_{i,j}) + (\bar{\alpha}_{i,1} + \bar{\alpha}_{i,J_i})(1 - \sum_{j=1}^{J_i} \alpha_{i,j}^k) \tag{22}$$

After measuring the rules, the index between the extended belief rule can be established through the Ball tree algorithm. In essential, the establishment of the Ball tree uses the divide and conquer method. In the beginning,

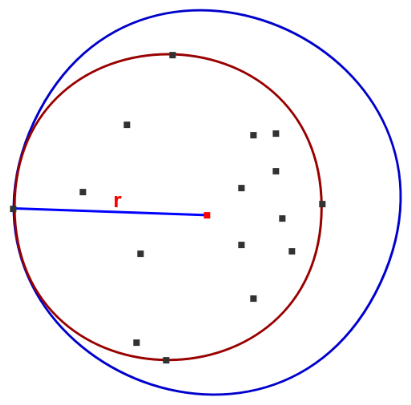

Figure 3: Ball tree radius selection

the Ball tree has only one root node, which contains all the rules in the ruleset. Then, the following steps proceeds as:

Step 1: Obtain the current node set which contains the center point and the Ball tree radius of all child nodes. The traditional way is to get the center point by finding the points in the node set. Then find the distance from the point farthest from the center point as the radius. However, the radius and the center point obtained by this method are not the best ones. A counter example is as illustrated in Figure 3: The blue big circle in the Figure 3 is the circle obtained by the traditional method, while the red small circle is actually the optimal one. Due to the distribution of the data set, the midpoint selection of the Ball tree can not find the position to make the radius the smallest. When the radius is too large, the efficiency of retrieving the Ball tree reduces. Therefore, after determining the midpoint and radius, we further optimize the selected midpoint by simulated annealing.

Step 2: Randomly select a belief rule in the ruleset of the current node, and consequently find the belief rule $A$ farthest from the rule as well as the rule $B$ farthest from point $A$.

Step 3: According to the farthest point pair $A$ and $B$ found, the rule assembly point is divided into the left and right subballs. The traditional method is to directly calculate the distance between the rule assembly point and one points within points $A$ and $B$. In order to achieve better classification results, a Ball tree is constructed through employing $k$-means and is shown with a lower tree height. In essential, point $A$ and point $B$ are used as the initial clustering points with the two types of clustering beings regarded as the left and right subtrees.

Repeat steps 1, 2, and 3 for the left and right subtrees until the number of rules of the left and right subtrees is less than a threshold $M$.

The detailed process of building BTDCS-EBRB can be seen in Algorithm 1. Through the divide and conquer method, a Ball tree is eventually constructed and a balanced binary tree structure index is accordingly established for the extended belief rules of the original disordered storage in the selected metric space.

*4.2. Rule retrieval of BTDCS-EBRB*

The retrieval strategy of our BTDCS-EBRB system is to query each point in the radius range for the input information following the constructed Ball tree index. More precisely, for the input data $X$, the algorithm goes through all the points in the extended belief rule that satisfy $d(R, X) \leq \theta$ in the Ball tree. Algorithm 2 formally describes the retrieval process for the Ball tree.

Step 1: Set a threshold $\theta$, which represents the radius of the input data hypersphere.

Step 2: Determine whether the current node is a leaf node. If so, the rule is to be activated if the metric distance between each point and $X$ in the ruleset of the current node is within the threshold $\theta$; Otherwise, go into the left and right son nodes for rules that could be activated according to whether the hypersphere of the left and right son nodes of the current node intersects with $X$.

Step 3: Repeat step 2 for the left and right son nodes that intersect $X$.

In step 1, the determination of the threshold radius is typically based on the characteristics of the data set. However, the threshold selection will be different because of the distribution of data in the dataset. It is clearly impossible to set each dataset manually, so it remains to automatically determine the threshold radius of input data according to different data sets. For the task, we use the minimum Ball tree radius on the resulting tree as the initial threshold radius of the input data, and retrieve the Ball tree accordingly. If the radius cannot make the input data and the Ball tree intersect or the $\sqrt{k}$ ($k$ is the size of the training data set) rules can be found on the Ball tree, to ensure the retrieval speed, the initial radius is multiplied. When the number of rules retrieved is more than $\sqrt{k}$, then the data is sorted only by the first $k$ rules. The formal layout of the method is as illustrated in Algorithm 3.

During retrieval of activation rule of the BTDCS-EBRB, we have $d \leq r_p + r_x$, where $d$ is the distance between the input data and the center of the

**Algorithm 2:** Query of BTDCS-EBRB

```
Input: Query vector($X := (x_1, x_2, x_T)$), Retrieval threshold($radius$),
       nodes on BallTree($curNode$)
Output: Set of rules to be activated($activatedRules$)
if distance between curNode.pivot and testQuery < radius then
    activatedRules.addAll(rules in curNode)
    return
end
if curNode is leaf node then
    for rule in curNode.ruleSet do
        dis := distance between rule and testQuery
        if dis < radius then
            activatedRules.add(rule)
            return
        end
    end
end
leftDis := distance between curNode.leftChild.pivot and
  testQuery
rightDis := distance between curNode.rightChild.pivot and
  testQuery
if leftDis-curNode.leftChild.radius¡radius then
    Query(X,radius,curNode.leftChild)
end
if rightDis-curNode.rightChild.radius¡radius then
    Query(X,radius,curNode.leftChild)
end
```

**Algorithm 3:** Retrieval of activation belief rule

```
**Input:** vercor($(X := (x_1, x_2, x_T))$))
**Output:** Set of rules to be activated($activatedRules$)
$radius$ := getTreeMinRadius()
**while** $activatedRules$ ¡ $K$ **do**
    $ruleList$ := Query($X$, $radius$, $rootofBallTree$)
    $radius := radius * 2$
**end**
sort($ruleList$)
$activatedRules$ := getSubArray($ruleList$, 0 , $K$)
return $activatedRules$
```

node of the Ball tree, $r_p$ is the radius of the Ball tree node, and $r_x$ is the threshold radius. If $d \leq r_p + r_x$ does not hold, we will not retrieve rules in the subtree of the node for improving the retrieval efficiency and accuracy of the BTDCS-EBRB system.

### *4.3. Runtime analysis*

The time complexity of our BTDCS-EBRB system is mainly composed by the construction and retrieval of the Ball tree index. Suppose there are $N$ rules, each rule has $M$ premise attributes, each premise attribute has $K$ reference values, and the result evaluates the number of ranks $R$. Then the time generated by each rule is $O(MK)$, so the time for generating the initial rule set is $O(NMK)$.

Assume that the generated Ball tree is a balanced binary tree. Then the height of the Ball tree is $\log_2 N$. Since the rule number of each layer of the Ball tree is $N$ in total, the calculation center and radius required for each layer are $O(2NM)$ and the calculation time required to separate the point set is $O(2NM + 2NM)$. Hence, the build time of the BallTree index is $O(NM \log_2 N)$.

The BTDCS-EBRB rule base reasoning can be divided into two parts: the Ball tree retrieval and the ER reasoning. During the retrieval process of the Ball tree, each iteration compares the center distance of the child nodes with the current node and consumes a runtime of $O(2M)$, resulting in a total retrieval time $O(2M \log_2 N)$. On the other hand, during adjusting the radius threshold radius, multiplication is performed until the number of found rules exceeds $S = \sqrt{N}$ in each iteration, where the rule of the first $S$ is sorted.

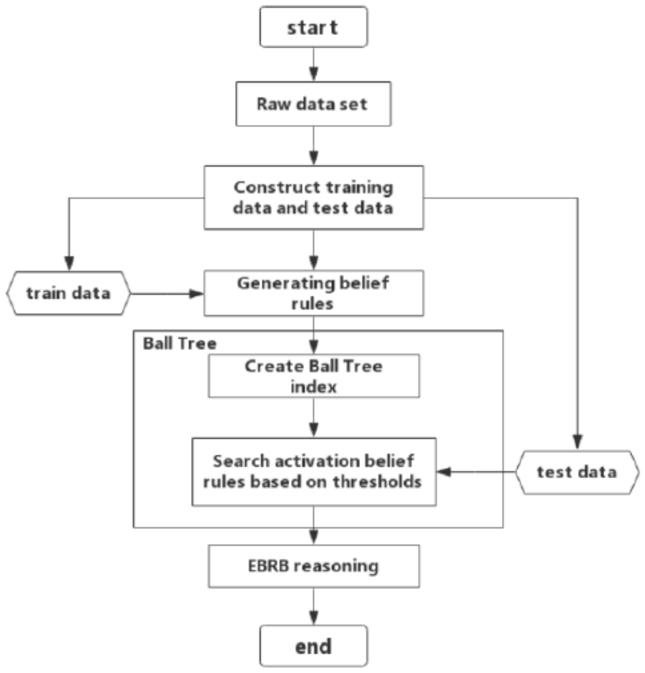

Figure 4: BTDCS-EBRB Flowchart

Assume the average number of doublings is $T$, then the total retrieval time is $O(2TM\log_2 N + SM\log_2 S)$.

Suppose there are $S$ rules for the final activation, then $O(MK)$ runtime is required to convert the input into a belief distribution, $O(SMK)$ is required to calculate the individual matching degree, and $O(SM)$ is required for the calculation of the rule weight. Therefore, the time for Ball tree-The reasoning of EBRB is $O(RS + S + R) = O(RS)$ in total.

To intuitively demonstrate the optimization effect of BTDCS-EBRB on EBRB with respect to the time complexity, we use the nonlinear function $f(x) = xcosx^2, 0 \le x \le 3$ to randomly generate 1500 training data sets and 30,000 test data sets to run EBRB and BTDCS-EBRB separately to calculate the runtime consumed by the two systems at different stages.

When constructing the rule, with the argument $x$ of the function as the premise attribute, we set 7 uniformly distributed different reference values $\{0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$ for x and 5 values for the system. For a different evaluation level, the utility value of each evaluation level is $\{-3.5, -2.5, 0, 2.0, 3.5\}$. Within the range of values of the independent variables, the $x$ values are uni-

Table 1: System reasoning time comparison

| Reasoning stage | EBRB/s | BTDCS-EBRB/s |
|---|---|---|
| Generation of belief rule | 0.364 | 0.364 |
| Construction of belief rule index | - | 0.011 |
| Retriev of belief rules | - | 0.327 |
| ER reasonings | 5.990 | 0.045 |
| Total time | 6.334 | 0.747 |

Table 2: Basic information of classification datasets.

| Names | No.of attributes | No.of classes | No.of samples |
|---|---|---|---|
| Banknote | 4 | 2 | 1372 |
| Iris | 4 | 3 | 150 |
| Mammographic | 5 | 2 | 830 |
| Knowledge | 5 | 4 | 403 |
| Vertebral | 6 | 3 | 310 |
| Seeds | 7 | 3 | 210 |
| Ecoli | 7 | 2 | 336 |
| Pima | 8 | 2 | 768 |
| Yeast | 8 | 10 | 1484 |
| Glass | 9 | 6 | 214 |
| Wine | 13 | 3 | 178 |
| Cancer | 30 | 2 | 569 |

formly generated at a certain interval and the corresponding $f(x)$ is calculated. As demonstrated in Table 1, our BTDCS-EBRB is about 36.86 times faster than the traditional methods. Moreover, when the size of the data set grows, the time gap becomes larger.

## 5. Numerical Simulation

In order to evaluate the effectiveness of our proposed method, three experiments were performed in this section. The first mainly aims to compare and analyze the height of the traditional $k$-center based Ball tree and our proposed $k$-means based Ball tree. The second is to investigate the threshold radius of the Ball tree for retrieving, and the third is to demonstrate the overall performance of the BTDCS-EBRB system via comparing with the existing methods including EBRB[12], its improved version by Yang et al.[29], and BK-EBRB[19]. The experiment used 12 common classification data of UCI, whose details are as depicted in Table 2.

The experimental was carried out on a platform of Windows 10 operating system with Intel® Core$^{TM}$ i5-7300HQ CPU @ 2. 50 GHz and 8GB ram. Our algorithms are implemented using C++ with G++ as the compiler in which -O2optimization is turned on. The main evaluation indicators involved in the experiment are as follows:

- Accuracy: The ratio of the number of correctly classified rules to the number of all rules.
- VRR: Visited rule rate, the ratio of the number of rules retrieved to the total number of rules when the system calculates the rule activation weight. The lower the VRR, the fewer rules are retrieved, and the higher the inference efficiency is.
- MAE: Mean absolute difference, the average of the errors between the system-inferred results and the actual results

## *5.1. Research on rule division method of Ball Tree*

In the Ball tree algorithm, the index structure of the rules during construction will affect the retrieval speed of the rules in the inference process. The traditional Ball tree construction algorithm divides the rules by a simple 2-center method, where the two center points are a pair of farthest points. For the classification of rules, only the distances between other points and the farthest point pairs are considered. In order to optimize the classification effect and get a better Ball tree index structure, we use 2-means method instead of 2-center. In order to reduce the number of iterations, the farthest point pair is also selected for the initial point pair in the experiment. The methods of clustering, adjusting the center of gravity, and re-clustering through several iterations eventually result in a better Ball tree index structure. In order to verify the effectiveness of this method, the text analyzes the effects of the 2-center method and the 2-means method through five commonly used UCI data sets.

As demonstrated in Table 3, the 2-means method has a higher accuracy and lower rule access rate on multiple data sets than 2-centers. When the tree is constructed, the classification result is adjusted by the 2-means method to

Table 3: Comparison of system effects under optimal conditions and radius adaptation

| Dataset | Aspects | 2-centers | 2-means |
|---|---|---|---|
| banknote | Tree length | 17.53 | 16.57 |
| | Accuracy(%) | 99.18 | 99.20 |
| | VRR(%) | 6.03 | 5.93 |
| vertebral | Tree length | 15.46 | 14.4 |
| | Accuracy(%) | 82.19 | 82.70 |
| | VRR(%) | 23.69 | 23.54 |
| pima | Tree length | 18.66 | 17.29 |
| | Accuracy(%) | 74.16 | 74.32 |
| | VRR(%) | 33.03 | 32.49 |
| wine | Tree length | 12.11 | 11.2 |
| | Accuracy(%) | 93.89 | 94.76 |
| | VRR(%) | 59.68 | 58.71 |
| cancer | Tree length | 12.42 | 10.88 |
| | Accuracy(%) | 95.60 | 96.00 |
| | VRR(%) | 21.60 | 21.29 |

reduce the height of the Ball tree. This indicates that a more balanced rule index library is constructed and the retrieval efficiency is improved.

## *5.2. Research on retrieval threshold*

Since the threshold radius is set to affect the retrieval efficiency of the system when retrieving for the Ball tree. To further understand the impact of threshold changes on system performance, this section evaluates the efficiency of the Ball tree EBRB system at different threshold radii through the three UCI datasets of cancer, ecoli, wine and pima.The numerical results are demonstrated as in Figure (5-8).

It can be seen from the Figure (5-8) that the change of the radius threshold has a great influence on the performance of the Ball tree-EBRB system. For different data sets, the change trend of the Ball tree during the retrieval is basically consistent. As the radius threshold grows, the accuracy of the Ball tree-EBRB system gradually increases. However, after reaching the optimal radius threshold, the accuracy begins to decrease as the threshold continues to grow. The VRR also gradually decreases when the radius threshold grows before reaching a minimum point, and gradually rises afterwards. This is

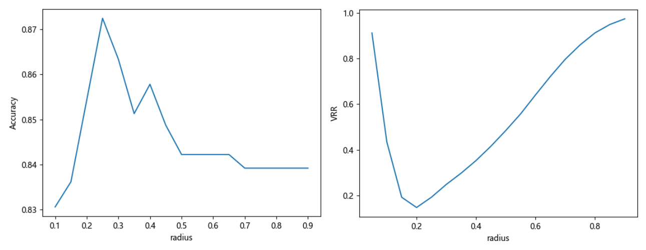

Figure 5: Experimental results of Ball tree-EBRB on ecoli

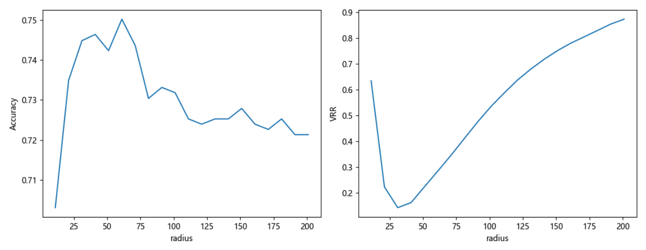

Figure 6: Experimental results of Ball tree-EBRB on pima

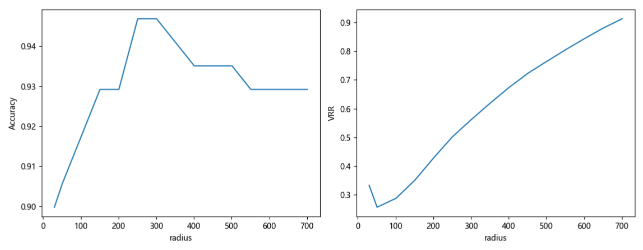

Figure 7: Experimental results of Ball tree-EBRB on wine

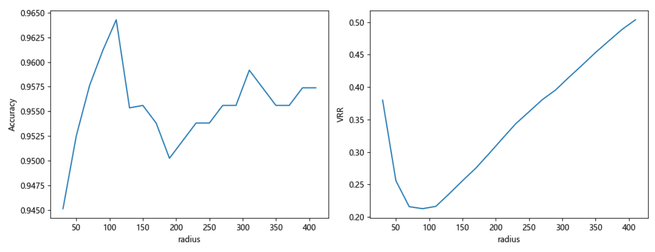

Figure 8: Experimental results of Ball tree-EBRB on cancer

Table 4: Comparison of system effects under optimal conditions and radius adaptation

| Dataset | Aspects | Optimal | Adaptive |
|---|---|---|---|
| ecoli | Accuracy(%) | 86.34 | 86.29 |
| | VRR(%) | 24.78 | 22.82 |
| pima | Accuracy(%) | 75.03 | 74.32 |
| | VRR(%) | 28.43 | 32.49 |
| wine | Accuracy(%) | 94.68 | 94.76 |
| | VRR(%) | 50.21 | 58.71 |
| cancer | Accuracy(%) | 96.43 | 96.00 |
| | VRR(%) | 21.16 | 21.29 |

mainly because when the radius threshold is getting smaller that it is even close to 0. In the case, the Ball tree-EBRB system follows the activation rules for the input data points in the radius, so that all the rules are used as the activation rules for subsequent reasoning. Because there are many rules involved in subsequent reasoning, it will undoubtedly affect the runtime of the system. On the other hand, too many rules bring more irrelevant rules to the reasoning, resulting in reduced accuracy of the system. So when the radius is too large, the Ball tree also has too many radical rules retrieved in the radius range, thereby reducing the running speed of the Ball tree-EBRB system and the accuracy.

For different datasets, because the characteristics of the dataset are different, the Ball tree-EBRB system cannot achieve the best retrieved results for all datasets based on the value of a radius threshold. In order to realize the generalization ability of the system on different data sets, we propose a method of threshold radius adaptation for retrieval, hereinafter referred to as BTDCS-EBRB system. According to this method, we run and compared the values of Accuracy against four more common data sets, ecoli, wine, cancer, and pima. The results are as demonstrated in Figures (5-8), where the analysis of Ball tree-EBRB is under optimal conditions and radius adaptation.

As shown in Table 4, the accuracy of the system was slightly compromised, i.e. about 1% $\sim$ 2%, comparing to the optimal value of the system based on the best threshold value obtained. This indicates that the adaptive method makes the system more generalized, while maintains the accuracy and time at a good level, reflecting its better generalization ability.

Table 5: Comparison of EBRB and BK-EBRB

| EBRB system | MAE | Reasoning time/s | Retrieved rule |
|---|---|---|---|
| EBRB | 0.236659 | 833.0000 | - |
| BTDCS-EBRB | 0.181532 | 0.6760 | 309069 |
| BK-EBRB(theta=0.4) | 0.231400 | - | 1432841 |

## *5.3. Comparative Experiment*

In order to further evaluate the speed optimization effect of BTDCS-EBRB, the BTDCS-EBRB system was compared with the methods proposed by EBRB[12], BK-EBRB[19], and the method proposed by Yang el al.[29].

Table 5 lists the performance of the EBRB system, the BK-EBRB system, and the BTDCS-EBRB system in the oil pipeline leakage simulation data set [24, 32]. The data pipeline of the oil pipeline is in the 2007 group data, according to the method of [4], a total of 1500 data are randomly selected from three time periods as training data according to a certain proportion, and a belief rule is generated. MAE, reasoning time, and number of retrieved rules are used as metrics.

As demonstrated in table 5, BTDCS-EBRB has better accuracy and lower number of retrieved rules than EBRB and BK-EBRB systems. This indicates we can improve the efficiency of EBRB system reasoning through making the EBRB system reflect the behavior of the system more accurately.

Table 6 demonstrates the effect of the BTDCS-EBRB system and the EBRB system proposed by Yang et al.[29] and DRA-EBRB system[1] on 11 classified data sets. In the experiment, the premise attributes of each data set are uniformly selected according to all the values of the respective attributes in the data set as the reference value. In order to reduce the error in the experiment, a $k$-fold cross-validation experiment(k-CV)[18] was introduced. The training and test sets are obtained by cross-folding by ten percents. The method used in the experiment was 10 cross-validation and averaging, and the evaluation indicators were: Accuracy, VRR $\pm$ standard deviation.

As shown in Table 6, the classification accuracy of the BTDCS-EBRBcan improve the classification accuracy on the Iris and Yeast dataset. The accuracy rate of several other classified data sets is also not large, and the difference between the other systems is generally about 1%. However, BTDCS-EBRB has greatly improved the retrieval speed of EBRB activation rules. In particular, for the high-dimensional 13-dimensional dataset Wine dataset, the VRR of the EBRB of the sphere tree is optimized by 27.05% and 13.68%, respectively, comparing with the EBRB of the KD tree and the EBRB of the BK tree. Notably, the improvement against Cancer data sets are 67.67% and 76.15%, respectively.

For k-d tree, there will be a dimensional explosion problem in high-dimensional data, so that most search results in high-dimensional space are unnecessary brute force search with efficiency even worse than exhaustive search. The analysis of the binary search tree found that the range search worst case time for a $k$-dimensional KD tree with $N$ nodes is $t_worst = O(kN^{1-\frac{1}{k}})$[9]. When $k$ is large, the k-d tree will degenerate into $O(n)$ algorithm. As depicted in Table 6, when the data dimension increases, the retrieval efficiency of the KSKDT algorithm is getting worse, while our algorithm still maintains good stability on high-dimensional data because each node is a hypersphere in the Ball tree. This suits the theoretical analysis that the time complexity of the bk tree retrieval is $N^{\alpha}(0 \leq a \leq 1)$, and the runtime of the Ball tree is $O(\log N)$. Where $N$ is the number of rules in the training set. In terms of complexity, the Ball tree has a good advantage.

## 6. Conclusion

Observing the inefficiency of retrieving unordered rules for activation in the EBRB system, this paper devised an optimized framework for indexing and retrieving the rules based on improved Ball trees. The indexing scheme uses improved Ball trees with the distances between the nodes indicating the similarity of the rules, such that the relationship between the rules is essentially embedded when storing the rules. Moreover, we proposed a self-adaptive algorithm for rule retrieval which finds an appropriate radius threshold and consequently enables faster retrieval with more relevant activated rules in the vicinity of the input data. Last but not the least, extensive numerical experiments were carried out to valuate the improved reasoning speed and inferencing accuracy of our BTDCS-EBRB system via comparing with other baselines including state-of-art approaches.

Table 6: Comparison of Our Approach Against Yang et al.

| Dataset | Aspects | DRA+WA | DRA+ER | EBRB | KSBKT+SBR | KSKDT+SBR | BTDCS |
|---|---|---|---|---|---|---|---|
| Banknote | Accuracy(%) | **99.70(1)** | **99.70(1)** | 96.74(6) | 99.58(3) | 99.58(3) | 99.20(5) |
| | VRR(%) | 100(4) | 100(4) | 100(4) | 22.81±2.00(3) | 8.23±0.05(2) | **5.93±0.99(1)** |
| Iris | Accuracy(%) | 95.63(2) | 95.50(3) | 95.33(4) | 95.20(5) | 95.20(5) | **95.99(1)** |
| | VRR(%) | 100(4) | 100(4) | 100(4) | 16.95±1.38(2) | 19.70±0.22(3) | **14.25±1.65(1)** |
| Mammographic | Accuracy(%) | 78.67(5) | 78.39(6) | 79.70(4) | **80.61(1)** | **80.61(1)** | 79.80(3) |
| | VRR(%) | 100(4) | 100(4) | 100(4) | 16.75±2.42(2) | 18.42±0.33(3) | **16.58±2.25(1)** |
| Knowledge | Accuracy(%) | 80.67(5) | 80.71(4) | 80.47(6) | **87.59(1)** | **87.59(1)** | 82.46(3) |
| | VRR(%) | 100(4) | 100(4) | 100(4) | 56.72±2.82(3) | **31.65±0.37(1)** | 38.82±3.40(2) |
| Vertebral | Accuracy(%) | 83.66(2) | **84.29(1)** | 72.94(4) | 76.39(5) | 76.39(5) | 82.71(3) |
| | VRR(%) | 100(4) | 100(4) | 100(4) | 40.14±5.41(3) | 31.45±0.40(2) | **23.54±4.03(1)** |
| Seeds | Accuracy(%) | 92.14(4) | 92.02(5) | 91.33(6) | 93.33(2) | **93.71(1)** | 92.20(3) |
| | VRR(%) | 100(4) | 100(4) | 100(4) | 37.73±2.70(3) | 30.39±0.27(2) | **17.41±2.90(1)** |
| Ecoli | Accuracy(%) | 83.75(5) | 83.76(4) | 81.16(6) | **86.93(1)** | **86.93(1)** | 86.28(3) |
| | VRR(%) | 100(4) | 100(4) | 100(4) | 42.01±6.13(3) | 34.01±0.42(2) | **22.82±4.30(1)** |
| Yeast | Accuracy(%) | 54.15(4) | 54.13(5) | 45.61(6) | 59.77(2) | 59.77(2) | **59.87(1)** |
| | VRR(%) | 100(4) | 100(4) | 100(4) | 57.94±3.36(3) | 46.85±0.24(2) | **18.48±1.64(1)** |
| Glass | Accuracy(%) | **70.26(1)** | 69.65(5) | 67.85(6) | 70.19(2) | 69.86(4) | 70.01(3) |
| | VRR(%) | 100(4) | 100(4) | 100(4) | 35.70±3.39(2) | 39.05±0.37(3) | **30.25±8.29(1)** |
| Wine | Accuracy(%) | 96.40(3) | 96.46(2) | 96.24(5) | 96.40(3) | **96.52(1)** | 94.76(6) |
| | VRR(%) | 100(4) | 100(4) | 100(4) | 72.39±10.83(2) | 80.48±0.49(3) | **58.71±8.58(1)** |
| Cancer | Accuracy(%) | 94.52(6) | 94.61(5) | 96.52(3) | **96.63(1)** | **96.63(1)** | 96.00(4) |
| | VRR(%) | 100(4) | 100(4) | 100(4) | 65.85±8.67(2) | 89.31±0.14(3) | **21.29±3.94(1)** |
| Average rank | Accuracy | 3.45(4) | 3.72(5) | 5.09(6) | 2.54(2) | **2.36(1)** | 3.18(3) |
| | VRR | 4(4) | 4(4) | 4(4) | 2.54(3) | 2.27(2) | **1.18(1)** |

**Acknowledgment**

The research was supported by National Nature Science Foundation of China (Nos 61773123, 61772005) and Natural Science Foundation of Fujian Province (Nos 2019J01647, 2017J01753).

## Reference

## References

[1] Alberto Calzada, Jun Liu, Hui Wang, and Anil Kashyap. A new dynamic rule activation method for extended belief rule-based systems. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):880–894, 2014.

[2] Alberto Calzada, Jun Liu, Hui Wang, Chris Nugent, and Luis Martinez. Application of a spatial intelligent decision system on self-rated health status estimation. *Journal of medical systems*, 39(11):138, 2015.

[3] Lei-Lei Chang, Zhi-Jie Zhou, Huchang Liao, Yu-Wang Chen, Xu Tan, and Francisco Herrera. Generic disjunctive belief-rule-base modeling, inferencing, and optimization. *IEEE Transactions on Fuzzy Systems*, 27(9):1866–1880, 2019.

[4] Yu-Wang Chen, Jian-Bo Yang, Dong-Ling Xu, Zhi-Jie Zhou, and Da-Wei Tang. Inference analysis and adaptive training for belief rule based systems. *Expert Systems with Applications*, 38(10):12845–12860, 2011.

[5] Chen Cheng, Liu Xiaoli, Wei Linfeng, Lin Longxin, and Wu Xiaofeng. Algorithm for k-anonymity based on ball-tree and projection area density partition. In *2019 14th International Conference on Computer Science & Education (ICCSE)*, pages 972–975. IEEE, 2019.

[6] Arthur P Dempster. A generalization of bayesian inference. *Journal of the Royal Statistical Society: Series B (Methodological)*, 30(2):205–232, 1968.

[7] Macarena Espinilla, Javier Medina, Alberto Calzada, Jun Liu, Luis Martínez, and Chris Nugent. Optimizing the configuration of an heterogeneous architecture of sensors for activity recognition, using the extended belief rule-based inference methodology. *Microprocessors and Microsystems*, 52:381–390, 2017.

[8] George J Klir and Bo Yuan. Fuzzy sets and fuzzy logic: theory and applications. *Upper Saddle River*, page 563, 1995.

[9] Der-Tsai Lee and CK Wong. Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees. *Acta Informatica*, 9(1):23–29, 1977.

[10] Gailing Li, Zhijie Zhou, Changhua Hu, Leilei Chang, Hongtao Zhang, and Chuanqiang Yu. An optimal safety assessment model for complex systems considering correlation and redundancy. *International Journal of Approximate Reasoning*, 104:38–56, 2019.

[11] Yan-Qing Lin, Yang-Geng Fu, Qun Su, Ying-Ming Wang, and Xiao-Ting Gong. A rule activation method for extended belief rule base with vp-tree and mvp-tree. *Journal of Intelligent & Fuzzy Systems*, 33(6):3695–3705, 2017.

[12] Jun Liu, Luis Martinez, Alberto Calzada, and Hui Wang. A novel belief rule base representation, generation and its inference methodology. *Knowledge-Based Systems*, 53:129–141, 2013.

[13] Rui Mao, Sheng Liu, Honglong Xu, Dian Zhang, and Daniel P Miranker. On data partitioning in tree structure metric-space indexes. In *International Conference on Database Systems for Advanced Applications*, pages 141–155. Springer, 2014.

[14] Andrew W Moore. The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 397–405. Morgan Kaufmann Publishers Inc., 2000.

[15] Stephen M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[16] Panayiotis Petousis, Arash Naeim, Ali Mosleh, and William Hsu. Evaluating the impact of uncertainty on risk prediction: Towards more robust prediction models. In *AMIA Annual Symposium Proceedings*, volume 2018, page 1461. American Medical Informatics Association, 2018.

[17] Giang Nguyen Thi Phuong, Huong Hoang Luong, Tai Huu Pham, and Hiep Xuan Huynh. A parallel algorithm for determining the communication radius of an automatic light trap based on balltree structure. In *2016 Eighth International Conference on Knowledge and Systems Engineering (KSE)*, pages 139–143. IEEE, 2016.

[18] Sarunas J Raudys and Anil K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (3):252–264, 1991.

[19] Q Su, LH Yang, YG Fu, and RY Yu. Structure optimization framework of extended belief rule based on bk-tree. *Journal of Frontiers of Computer Science and Technology*, 10(2):257–267, 2016.

[20] Ron Sun. Robust reasoning: integrating rule-based and similarity-based reasoning. *Artificial Intelligence*, 75(2):241–295, 1995.

[21] Weiguo Wan and Hyo Jong Lee. Deep feature representation and balltree for face sketch recognition. *International Journal of System Assurance Engineering and Management*, pages 1–6, 2019.

[22] Ying-Ming Wang, Jian-Bo Yang, and Dong-Ling Xu. Environmental impact assessment using the evidential reasoning approach. *European Journal of Operational Research*, 174(3):1885–1913, 2006.

[23] Ying-Ming Wang, Fei-Fei Ye, and Long-Hao Yang. Extended belief rule based system with joint learning for environmental governance cost prediction. *Ecological Indicators*, 111:106070, 2020.

[24] Dong-Ling Xu, Jun Liu, Jian-Bo Yang, Guo-Ping Liu, Jin Wang, Ian Jenkinson, and Jun Ren. Inference and learning methodology of belief-rule-based expert system for pipeline leak detection. *Expert Systems with Applications*, 32(1):103–113, 2007.

[25] Jian-Bo Yang and Dong-Ling Xu. On the evidential reasoning algorithm for multiple attribute decision analysis under uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 32(3):289–304, 2002.

[26] Long-Hao Yang, Jun Liu, Ying-Ming Wang, and Luis Martínez. A micro-extended belief rule-based system for big data multiclass classification problems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[27] Long-Hao Yang, Jun Liu, Ying-Ming Wang, and Luis Martínez. New activation weight calculation and parameter optimization for extended belief rule-based system based on sensitivity analysis. *Knowledge and Information Systems*, 60(2):837–878, 2019.

[28] Long-Hao Yang, Ying-Ming Wang, Yi-Xin Lan, Lei Chen, and Yang-Geng Fu. A data envelopment analysis (dea)-based method for rule reduction in extended belief-rule-based systems. *Knowledge-Based Systems*, 123:174–187, 2017.

[29] Long-Hao Yang, Ying-Ming Wang, Qun Su, Yang-Geng Fu, and Kwai-Sang Chin. Multi-attribute search framework for optimizing extended belief rule-based systems. *Information Sciences*, 370:159–183, 2016.

[30] Rui-Yin Yu, Long-Hao Yang, and YG Fu. Data driven construction and inference methodology of belief rule-base. *Journal of Computer Applications*, 34(8):2155–2160, 2014.

[31] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[32] Zhi-Jie Zhou, Chang-Hua Hu, Jian-Bo Yang, Dong-Ling Xu, and Dong-Hua Zhou. Online updating belief rule based system for pipeline leak detection under expert intervention. *Expert Systems with Applications*, 36(4):7700–7709, 2009.

[33] Zhi-Jie Zhou, Guan-Yu Hu, Chang-Hua Hu, Cheng-Lin Wen, and Lei-Lei Chang. A survey of belief rule-base expert system. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.