An Improved Cellular Automata-Based Classifier with Soft Decision

Pattapon Wanna, Sartra Wongthanavasu

Department of Computer Science, Khon Kaen University, Thailand
pattaponw@kkumail.com, wongsar@kku.ac.th

Abstract

Classification has been successfully applying in problems in a variety of fields, such as science, business, engineering, and industry. Unfortunately, the classifier coping with nonconforming binary patterns are rare. To deal with nonconforming pattern in binary Cellular Automata-based Classifier (CAC) had been proposed. However, CAC faces several limitations that need to improve. First, the rule ordering process in CAC which used Genetic Algorithm (GA) is unable to handle high dimensional complex problems. Second, finding decision boundaries is quite rough when dealing with ambiguous data. To deal with these problems, therefore, we propose a new classifier, called Cellular Automata-Based Classifier with Soft Decision (CAS). We replace the GA with the promising optimization algorithm, called Butterfly Optimization, for the rule ordering process. Subsequently, we improve the classification performance by augmenting a Soft-Decision step. This Soft-Decision step uses the pruning method to create a soft decision table, which efficiently serves for filtering useless data. Finally, to verify the classification performance of the proposed method, ten datasets consisting of conforming and nonconforming patterns are experimented in comparison with the promising classifiers including CAC, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Naïve Bayes, and Deep Learning using K-fold cross-validation. In this regard, CAS provides the promising results.

Keywords: Cellular automata, Classification, Soft decision, Pruning, Butterfly optimization

1 Introduction

Classification is an important technique in pattern recognition and data mining [41]. There are several popular classification techniques, such as Support Vector Machine (SVM) [13, 60], Naive Bayes [3, 31], K-nearest neighbor [20, 36], Decision Tree [49], Random Forest [11-12], Deep learning [4], and so forth. Nowadays, the problems in classification have dramatically increased the variety and complexity of the problems. For example, to deal with multi-class classification problem, additional techniques such as probabilistic rule lists and the minimum description length (MDL) principle [38], Divide and Conquer [16], and Multiple Empirical Kernel Learning (MEKL) [51] are used. Moreover, Ordinal classification [47], a specific case of multi-class classification that has a natural sequence on a set of class labels, audio classification [58], Hyperspectral [10] and Fabric [59] image classification, including video classification [40] method were proposed, including solutions for classification of binary data, for example, data with the format 0 or 1, Yes or No, etc. Research focuses on managing this type of data, such as [17, 39].

On the other hand, Cellular Automata (CA) is a model that tends to be successful in applying to advanced classification research. There are some promising classifiers based on cellular automata dealing with classification in complicated problems. Two-dimensional cellular automata are used in research for image analysis and classification. [43] proposed a conceptual framework for two image processing methods to improve brain tumor segmentation: image transformation and segmentation algorithm. To cope with ambiguous tumor boundaries, it is not only can extract extracts from the spectral-spatial properties of HSIs automatically but also restrict training samples using different spectral sizes and spatial sizes. In 2015, [14] proposed the classifier with the concept of corrosion modelling and cellular automata to generate a texture descriptor, dealing with synthetic and natural texture images classification tasks. The results show that the proposed texture indicator is useful in classifying the surface according to the LLNA high success rate obtained in all cases. In addition, solving the density problem based on cellulara automata in both one-dimension with expand the neightbors of the current cell [27], Fixed-Length technique [15] and two-dimensional [53] is challenging.

As stated previously, to solve the gap of classifiers based on cellular automata, the researcher proposed the Cellular Automata-Based Learning Method for Classification (CAC) [55]; this classifier is capable of implementing the conforming and non-conforming

*Corresponding Author: Sartra Wongthanavasu; E-mail: wongsar@kku.ac.th

DOI: 10.3966/160792642020112106012

patterns in binary data. It is pattern classifiers based on elementary cellular automata. CAC creates two rules matrices using the "Rule Ordering process" using the Genetic algorithm (GA). The most crucial part of this classifier is the cellular automata rules represented by rule matrix. Rules matrices are divided into both positive and negative sides, then classify the class of data by using Decision function CAC consists of two rule matrices and provides binary classification. For this reason, CAC using decision directed acyclic graph (DDAG) [2, 46] structure to deal with multi-class classification problem.

The main idea of this research is how to improve the classification capability of the CAC classifier. To cope with the improvement, we classify a wide range of classifiers into two groups: hard classifier and soft classifier [33]. In general, a soft classification rule generally estimates the class conditional probabilities explicitly and then makes the class prediction based on the most considerable estimated probability. In contrast, hard classification bypasses the requirement of class probability estimation and directly estimates the classification boundary. Typical soft classifiers include some traditional distribution-based likelihood approaches, such as Naive Bayes and K-nearest neighbors. On the other hand, some margin-based approaches, such as SVM, especially CAC. Generally, distributional assumption-free belongs to the class of hard classification methods. There are some research using the Soft and Hard classification to improve classifiers performance, for example, in 2011, a margin-based classifier, including both hard and soft classifiers, called Large-margin unified machines(LUMs), which covers a broad range of margin-based classifiers, including both hard and soft ones. By offering a natural bridge from soft to hard classification, the LUM provides a unified algorithm to fit various classifiers and hence a convenient platform to compare hard and soft classification. As a result, it can also use as a probability estimation technique for hard classifiers such as the SVM. According to [28], Lee and Kim proposed an overlap-sensitive margin (OSM) classifier based on a modified fuzzy support vector machine and k-nearest neighbor algorithm to address imbalanced and overlapping data sets. The main idea of the proposed OSM classifier is to separate the data space into soft and hard-overlap regions using the modified fuzzy support vector machine algorithm.

According to CAC that uses the concept of two-class classification like SVM, and the classification performance is depended on elementary cellular automata rule matrices, it still faces two problems. Firstly, the data with ambiguity impact the rule ordering process, which is a process that converts data pattern to rules vectors using a configuration of Cellular Automata. The initial rules matrixes are many duplicates in the same location. This problem is known as a collision problem; the classification used boundary finding method is not useful for this problem. Some researchers try to solve collision problems using the optimization algorithm with numerical instead of discrete binary variables [18], changing particle directions [48]. Secondly, typically, the classifier has found that when data have a higher dimension, it will affect the efficiency of the classifier [8]. In this regard, CAC faces this problem when dealing with high dimensional data. CAC's rule ordering process using GA does not address the best solution and has got low classification accuracy when dealing with high dimensional data. GA has limitation when dealing with high dimensional problems [26], especially for premature convergence and falling into a local optimum [45]. We want to find a new, efficient optimization algorithm. One of these is butterfly optimization, which has better performance than traditional optimization and has been improved for use in a variety of applications, such as feature selection [6, 57] and improve BOA by using mutualism scheme [42].

In this research, we have presented a highly efficient classification, called Cellular Automata-Based Classifier with Soft Decision (CAS), for solving problems faced in CAC. It reduce overfitting while rule ordering process and improves classification accuracy by using the Pruning method based on Soft decision based idea. Then implemented a Butterflies Optimization Algorithm (BOA) instead of GA in the rule ordering process, to deal with severe complexity problems. We validate the CAS in comparison with the state-of-the-art algorithms using ten UCI datasets.

The rest of the paper is organized as follows: In section 2, we introduce related work on Pattern Classifier Based on Decision Support Elementary Cellular Automata, a Butterfly Optimization Algorithm, Pruning method, and Principle component analysis for data virtualization. In section 3, we elaborate on our proposed CAS approach. In section 4, the empirical analysis and the comparison of experimental results are presented. Section 5 concludes our work.

2 Related Work

2.1 Pattern Classification Based on Decision Support Elementary Cellular Automata

Cellular Automata are systems evolving on lattices according to a local transition function [34]. It evolves through several discrete time steps according to a set of rules based on the states of neighboring cells. The rules are applied iteratively for as many time steps as desired. In 1983, S. Wolfram proposed the simplest type of cellular automata, called "Elementary Cellular Automata". It is a binary nearest-neighbor, one-dimensional automaton.

Elementary Cellular Automata (ECA) consists of two possible status groups (0 or 1), with the pattern of cells arranged in one dimension. A next state Q_i^{t+1} for the i^{th} cell is considered from its one nearest neighbor's local transition function $f(Q_{i-1}^t, Q_i^t, Q_{i+1}^t)$ of the present state Q_i^t For simplicity, a next state of n cells ECA is represented by a matrix *R* given following

$$R = (a_{ij})_{m\times n} \tag{1}$$

A rule matrix *R* consist of a set of inputs and solutions *m* rows and *n* columns, where *m* is bits pattern and $n \in I^+$, a_{ij} represents the members that are in row *i* and column *j* of the matrix. Thus, pattern classifiers based on the evolving structure of ECA is defined as follow.

$$Q^{t+1} = (R, Q^t) \tag{2}$$

Let *R* be a $|n \times 8|$ matrix representing the next state for *n* cells ECA, called a rule matrix. And element of the matrix, $a_{ij} \in \{0, 1\}$ is the next state i^{th} for the cell where it is nearest neighbors, (Q_{i-1}^t , Q_i^t , Q_{i+1}^t) is decode in decimal equal to *j*, *j* = 0 to 7. A general form of evolving ECA in a form of *R* is defined as following equation.

$$Q^{t+1} = \begin{cases} (R, Q^t),\ if\ Q^t \in Y \\ Q^t\ and\ stop, otherwise \end{cases} \tag{3}$$

In 2016, the researcher proposed a novel classifier based on cellular automata model, called Cellular Automata-based Classifier (CAC) was proposed. In Figure 1, it developed based on a new kind of one-dimensional cellular automata (ECA), called Decision Support Elementary Cellular Automata (DS-ECA). It is the elementary cellular automata with capability to choose a proper rule matrix for changing the state. It comprises two rule matrices (R^+ and R^-) and a decision function $f(Q_{R^+}^{t+1}, Q_{R^-}^{t+1}) \in \{-1, 1\}$. The $f(Q_{R^+}^{t+1}, Q_{R^-}^{t+1})$ is a sign function using $(Q_{R^+}^{t+1}, Q_{R^-}^{t+1})$ as parameters. The function is given in (4).

$$f(Q_{R^+}^{t+1}, Q_{R^-}^{t+1}) = \operatorname{sgn}\left(\sum_{i=0}^{n-1}(Q_{R^+}^{t+1} - Q_{R^-}^{t+1})\right) \tag{4}$$

where $Q_{R^+}^{t+1}$ and $Q_{R^-}^{t+1}$ are a next state generated by the rule matrices R^+ and R^- . $Q_{R^+}^{t+1}$ and $Q_{R^-}^{t+1}$ are the i^{th} cells of n bits from $Q_{R^+}^{t+1}$ and $Q_{R^-}^{t+1}$, respectively.

The two next states $Q_{R^+}^{t+1}$ and $Q_{R^-}^{t+1}$, are generated as follows:

$$Q_{R^+}^{t+1} = \left(R^+, Q^t\right) \tag{5}$$

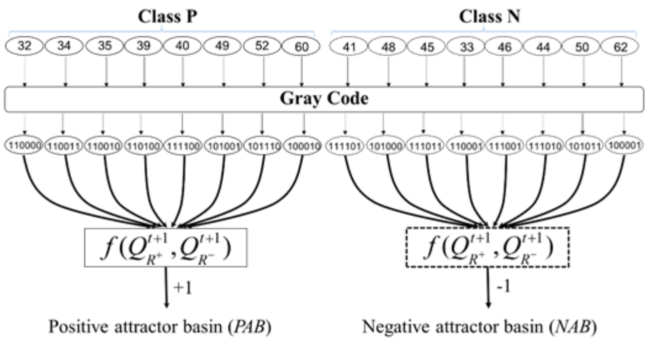


Figure 1. CAC with two attractor basins

$$Q_{R^-}^{t+1} = \left(R^-, Q^t\right) \tag{6}$$

Example 1. Suppose binary number '110011' is a 6-cell ECA with null boundary condition [1] and rule matrices R^+ and R^- ; the classification task could be processed by following steps:

$$R^+ = \begin{array}{c|cccccccc|} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 5 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 6 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

$$R^- = \begin{array}{c|cccccccc|} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 3 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 4 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 5 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}$$

Step 1: The first step, choosing a null boundary condition to define the ECA configuration, the first feature must add '0' on the left then choose left and right neighbor with itself.

$$\boxed{\underline{0}\ 1\ 1}\ 0\ 0\ 1\ 1$$

The first feature of the pattern matching with ECA configuration '011' then $\left[R\right]_{1,4}^{+} = 1$ and $\left[R\right]_{1,4}^{-} = 1$ also.

Step 2: Continue to choose left and right neighbors of the second feature.

$$\underline{0}\boxed{1\ 1\ 0}\ 0\ 1\ 1$$

The second feature of the pattern maching with ECA config-uration '110' then $\left[R\right]_{2,7}^{+} = 1$ and $\left[R\right]_{2,7}^{-} = 0$ Subsequently, repete step 2 for remaining features until reaching the last feature.

$$\underline{0}\;1\;\boxed{1\;0\;0}\;1\;1$$

The thirth feature, ECA configuration is '100' then $\left[R\right]^{+}_{3,5}=1$ and $\left[R\right]^{-}_{3,5}=0$.

$$\underline{0}\;1\;1\;\boxed{0\;0\;1}\;1$$

The forth feature, ECA configuration is '001' then $\left[R\right]^{+}_{4,2}=1$ and $\left[R\right]^{-}_{4,2}=0$.

$$\underline{0}\;1\;1\;0\;\boxed{0\;1\;1}$$

The fifth feature, ECA configuration is '011' $\left[R\right]^{+}_{5,4}=0$ and $\left[R\right]^{-}_{5,4}=1$.

Step 3: This step is similar to the step, but add '0' to be a right neighbor for the last feature.

$$\underline{0}\;1\;1\;0\;0\;\boxed{1\;1\;\underline{0}}$$

The ECA configuration of the last feature is '110' is maching with $\left[R\right]^{+}_{6,7}=0$ and $\left[R\right]^{-}_{6,7}=1$.

Step 4: Define the class of binary pattern '110011' by using (4)

$$f(Q^{t+1}_{R^{+}},Q^{t+1}_{R^{-}})=\operatorname{sgn}\begin{pmatrix}\left(\left[R\right]^{+}_{1,4}-\left[R\right]^{-}_{1,4}\right)+\left(\left[R\right]^{+}_{2,7}-\left[R\right]^{-}_{2,7}\right)\\+\left(\left[R\right]^{+}_{3,5}-\left[R\right]^{-}_{3,5}\right)+\left(\left[R\right]^{+}_{4,2}-\left[R\right]^{-}_{4,2}\right)\\+\left(\left[R\right]^{+}_{5,4}-\left[R\right]^{-}_{5,4}\right)+\left(\left[R\right]^{+}_{6,7}-\left[R\right]^{-}_{6,7}\right)\end{pmatrix}$$

Replace variables with the values from step 1-3:

$$f(Q^{t+1}_{R^{+}},Q^{t+1}_{R^{-}})=\operatorname{sgn}\begin{pmatrix}(1-1)+(1-0)+(1-0)+(1-0)\\+(0-1)+(0-1)\end{pmatrix}$$

Then calculate sign function:

$$f(Q^{t+1}_{R^{+}},Q^{t+1}_{R^{-}})=+1$$

From the answer is +1, let be known that binary number 110011 is in the positive class and the next step should generate by using (5).

The rule matrices are the performance for classifying patterns. In this respect, a Genetic Algorithm (GA) use to order the rules arriving at R^{+} and R^{-} for classification.

In general, GA cannot cope with the complicated problem with high dimensional space to arrive at the close-to-optimal solutions. Hence, it is an appropriate and challenging problem to improve the performance of CAC using promising advanced optimization techniques. Furthermore, the classifier uses the basis of finding boundaries to classify classes in certain types of data is not enough to solve a difficult problem. From a literature review, the amendment to the limitations of this classifier is to use class classifications with probability calculations from some method to deal with areas that are sensitive classifying.

2.2 Butterfly Optimization Algorithm (BOA)

The butterfly optimization algorithm (BOA) is a novel meta-heuristic algorithm that is inspired by the butterfly's feeding behavior. The effectiveness of BOA depends on the prob-ability parameter, which determines whether the butterfly must move towards the best butterfly of the population or perform a random search [7, 30]. The structure principally relies upon the butterfly prey technique, which utilizes smell acknowledgment to decide the area of nourishment or mating sets. The entire idea of discovery and handling relies upon three critical conditions: the fragrance (*f*), sensory exposure (*c*), stimulus intensity (*I*) and power (*a*). Utilizing these ideas, in BOA, the scent is defined as a component of the physical power of boost as follow:

$$f = cI^{a} \tag{7}$$

where *f* is the size of perfume recognition, namely the strength of the smell recognition of other butterflies, *c* is the sensory receptor, *I* is a stimulating force and a is the exponent power depending on the modality. There are three phases in BOA: (1) Initialization phase, (2) Iteration phase and (3) Finally In the first BOA operation, each time will then perform a recursive search, and in the final phase the algorithm will eventually be terminated when the best solution is found. In the initial step, the algorithm determines the problem-solving area and the purpose of the function — besides, the parameters used in the BOA set. The position of the butterfly will be generated randomly in the search area with their perfume values and suitability calculated and stored. This progress will include the initial phase and the recursion phase calculation at a later stage. The second step of the algorithm, such as the looping phase, multiple iterations, is performed by an algorithm. In each iteration, all butterflies in the solution area will move to a new location and then evaluate the suitability. The first algorithm will calculate the suitability of all butterflies in different positions in the solution area. Then, these butterflies will create fragrances by position using equation (7). There are two essential steps in the algorithm, such as local search procedures and global search procedures. In the global search process, butterflies move to the most appropriate butterfly g answer, which can be displayed using equation (8).

$$x_i^{t+1}=x_i^{t}+\left(r^{2}\times g*-x_i^{t}\right)\times f_i \tag{8}$$

where x_i^{t+1} is the solution vector x_i for i^{th} butterfly in iteration number *t*. Here, *g* represents the current best solution found among all the solutions in current iteration. Fragrance of i^{th} butterfly is represented by f_i and *r* is a random number in [0, 1]. Local search phase can be represented as

$$x_i^{t+1} = x_i^t + \left(r^2 \times x_j^t - x_k^t\right) \times f_i \tag{9}$$

where x_i^t and x_k^t are j^{th} and k^{th} butterflies from the solution space. If x_j^t and x_k^t belongs to the same swarm and r is a random number in [0, 1], then equation (9) becomes a local random walk. Finding food mates and mating by butterflies can occur both locally and globally. Searching for food may be important in the overall mating or butterfly search activity, considering the physical proximity and other factors such as wind, rain, weather, temperature, etc. So, a switch probability p is used in BOA to switch between common global searches to intensive local search. Criteria for stopping can determine in many ways, such as the maximum CPU time used, the maximum number of iterations, the maximum number of repetitions without improvement, the error rate, or other appropriate criteria when summarizing the repetition process. The algorithm will export the best solution that meets the most appropriate. The three steps above comprise a complete algorithm of butterfly optimization algorithms describes in the "Algorithm 1".

Algorithm 1: Butterfly Optimization Algorithm

```
input  : $n, I, a, p, c$
output: Best $bf$
1  Initialization;
2  Objective function $f(x), x = (x_1, x_2, \ldots, x_{dim})$,
     $dim$=number of dimensions;
3  Generate initialize population of n Butterflies
     $x_i (i = 1, 2, \ldots, n)$;
4  Stimulus Intensity $I_i$ at $x_i$ is determined by $f(x_i)$;
5  Define sensor modality $c$, power exponent $a$ and
     switch probability $p$;
6  LOOP Process :
7  while stopping criteria not met do
8      instructions;
9      foreach butterfly $bf$ in population do
10         Calculate fragrance for $bf$ using equation (7);
11     end
12     Find the best $bf$;
13     foreach butterfly $bf$ in population do
14         Generate a random number $r$ from [0, 1];
15         if $r < p$ then
16             Move towards best butterfly/solution
                 using equation (8);
17         else
18             Move randomly using equation (9)3;
19         end
20     end
21     Update the value of $a$
22 end
23 return best $bf$
```

2.3 Pruning Method

Pruning method is a technique in machine learning that reduces the computational complexity of classifier and aim to improve classification accuracy. In other words, the pruning process uses to prevent overfitting of machine learning [56]. Pruning methods are widely used in machine learning, especially in classification techniques, for example decision tree [19], deep learning [32]. In the case of collision problems. Due to a large number of useless classification elements causing the problem to be solved by defining only the scope that is not good enough. A popular method to eliminate such problems is the pruning method [5, 50].

Typically, pruning consists of 2 steps: (1) evaluating the efficiency in all elements then in the process (2) Eliminate the most useless elements, which in the beginning when cutting out the branches, will increase the accuracy and cut until the accuracy of the classification is lower. But the process to finding the most suitable value for this method is also a challenge for research. In research [24] has been tested using GA to calculate the most suitable value for pruning. The result is that it can reduce the computation time and get better results than normal cuttings.

In general, the pruning method is used to cut decision branches that cause poor decision-making efficiency. Based on this concept, we have implemented a pruning technique to improve the rule metrics used to enhance classification by eliminating values that cause poor classification efficiency in the rule metrics.

2.4 Principal Component Analysis (PCA)

Generally, a dataset cannot be able to plot and observed the distribution if the number of attributes is higher than three. Principal component analysis (PCA) [21] is a widely used dimensionality reduction technique in data analysis. It reduces high dimensional features to low dimension and maintains distances between data points as much as possible. Finally, the most significant first two or three eigenvectors choose to plot the graph.

3 Proposed Method

This research proposed an efficient classifier-based cellular automaton, called Cellular Automata-Based Classifier with Soft Decision (CAS). CAS shown in Algorithm 2 aims to improve the performance of the Cellular Automata-based Classifier (CAC) algorithm. As a result that CAC faced a classification accuracy problem when using just the boundary to identify the class and rule ordering by using the Genetic Algorithm (GA) cannot crop high dimensional problems. This research gets rid of such a limitation by applying Soft Decision (Pruning method) while Rule ordering process and implement Butterfly Optimization Algorithm (BOA) instead of Genetic Algorithm illustrate in Figure 2.

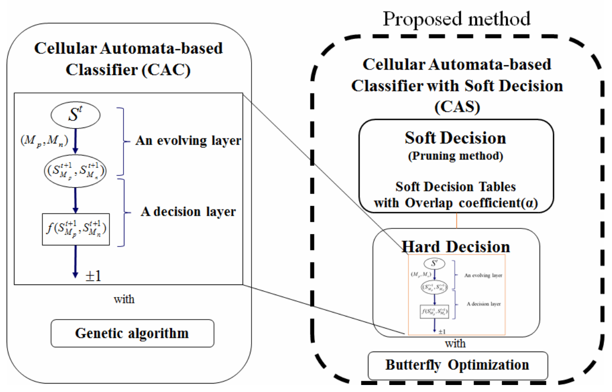

Proposed method
Cellular Automata-based Classifier (CAC)
An evolving layer
A decision layer
±1
with
Genetic algorithm
Cellular Automata-based Classifier with Soft Decision (CAS)
Soft Decision
(Pruning method)
Soft Decision Tables with Overlap coefficient(α)
Hard Decision
with
Butterfly Optimization

Figure 2. CAS overview

The initial matrix creation process uses the same as the original method, which defines the traditional boundary finding that focuses on only the boundary of the class into the hard decision section. Then, adding the Soft decision, which is a calculation of the percentage of the ability to classify each element of the rule matrix to eliminate the useless position of the rule matrices. The soft decision consists of 2 steps: the first step is to measure the ability of each element of the initial matrices before being introduced into the appropriate elemental valuation process by creating the Soft Decision table. It then leads to the optimization process, which is the endpoint of eliminating useless elements. This process occurs along with the determination of the boundary lines of the hard decision obtained from the butterfly optimization algorithm. Next, after obtaining the boundaries and points suitable for soft decision making, it will get rid of useless points with the value obtained above in the process of the pruning process. Finally, the rule synthesis process will be the last step in creating the rule matrix for data classification, as shown in Figure 3.

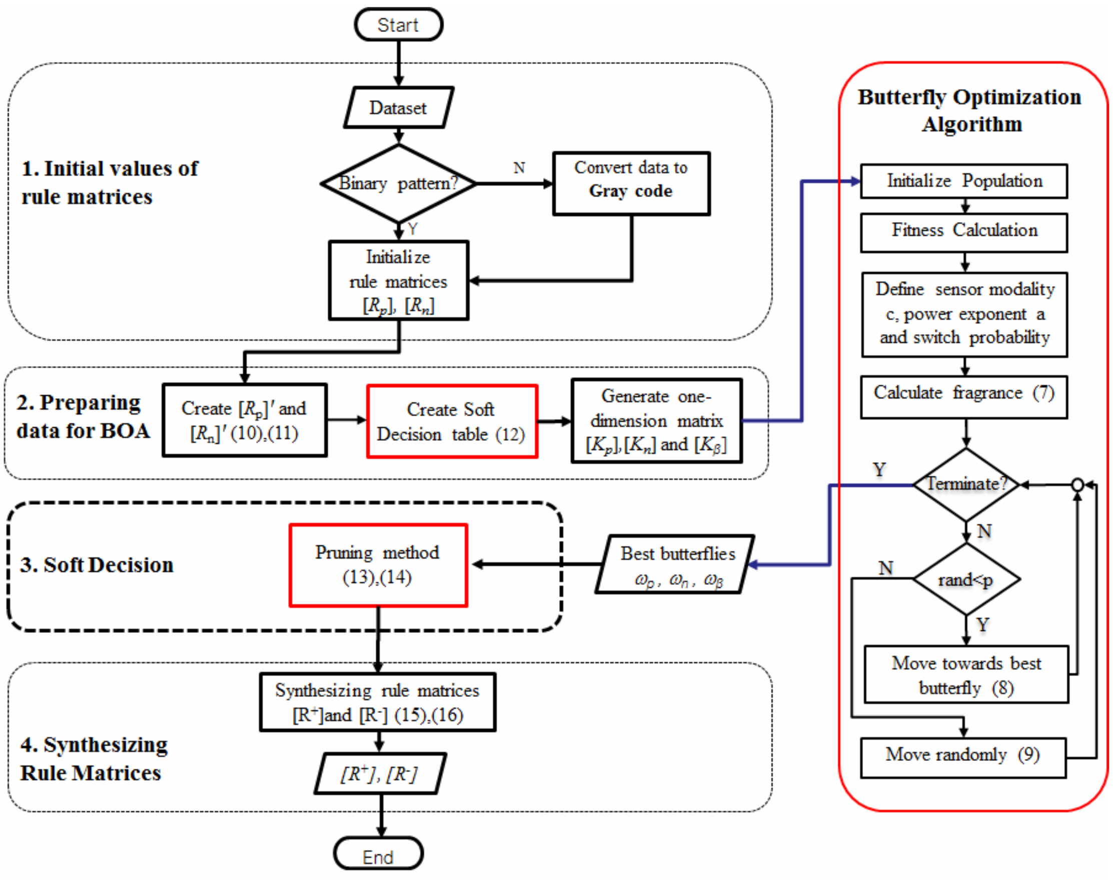

Start
Dataset
1. Initial values of rule matrices
Binary pattern?
N
Y
Convert data to Gray code
Initialize rule matrices [Rp], [Rn]
2. Preparing data for BOA
Create [Rp]' and [Rn]' (10),(11)
Create Soft Decision table (12)
Generate one-dimension matrix [Kp],[Kn] and [Kβ]
3. Soft Decision
Pruning method (13),(14)
Best butterflies ωp, ωn, ωβ
4. Synthesizing Rule Matrices
Synthesizing rule matrices [R+]and [R-] (15),(16)
[R+], [R-]
End
Butterfly Optimization Algorithm
Initialize Population
Fitness Calculation
Define sensor modality c, power exponent a and switch probability
Calculate fragrance (7)
Terminate?
Y
N
rand<p
N
Y
Move towards best butterfly (8)
Move randomly (9)

Figure 3. CAS rule ordering process

3.1 Initial Values of Rule Matrices

The Rule ordering process starts with converting the input to binary with Gray code [22, 35, 44] encoder if input data is not binary data. By using null boundary condition (First left neighbor is 0, last right neighbor is 0). The initial values of the matrices R_p and R_n are created by counting the number of patterns from an attractor basin (PAB or NAB) corresponding to the matrix. That is, and element of the matrix R_p in the i^{th} ($i = 0, 1, 2, \ldots, n$-1) row and the j^{th} ($j = 0, 1, 2, \ldots, n$-1) column is the number of patterns from PAB in which the nearest neighbors (Q_{i-1}^t, Q_i^t, Q_{i+1}^t) for the i^{th} cell decoded to decimal must be equal to j. Similarly, an element of the matrix R_n is formulated by NAB.

Example 2. Set up a training data(D) of 6 bits pat-tern for CAS is (110011,+1), (101100,+1), (111100,+1), (101001,+1), (100010,+1), (110000,+1), (110100,+1), (110010,+1), (111101,-1), (101000,-1), (111011,-1), (110001,-1), (111001,-1), (111010,-1), (101011,-1), (100001,-1). Set +1 and -1 is the class label of PAB and NAB, respectively. That mean PAB={110011, 101100, 111100, 101001, 100010, 110000, 110100, 110010} and NAB ={111101, 101000, 111011, 110001, 111001, 111010, 101011, 100001}. Subsequently, the initial rule matrices R_p and R_n are created as follow.

$$R^{+} = \begin{array}{c|cccccccc} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ \hline 1 & 0 & 0 & 3 & 5 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 1 & 2 & 4 & 1 \\ 3 & 1 & 0 & 1 & 1 & 3 & 1 & 0 & 1 \\ 4 & 1 & 3 & 1 & 0 & 1 & 0 & 1 & 1 \\ 5 & 1 & 1 & 2 & 1 & 2 & 0 & 1 & 0 \\ 6 & 3 & 0 & 1 & 0 & 3 & 0 & 1 & 0 \end{array}$$

$$R^{-} = \begin{array}{c|cccccccc} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ \hline 1 & 0 & 0 & 3 & 5 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 4 \\ 3 & 1 & 0 & 2 & 0 & 1 & 0 & 3 & 1 \\ 4 & 2 & 0 & 0 & 0 & 2 & 3 & 1 & 0 \\ 5 & 1 & 3 & 1 & 2 & 0 & 1 & 0 & 0 \\ 6 & 1 & 0 & 4 & 0 & 1 & 0 & 2 & 0 \end{array}$$

3.2 Preparing Data for BOA

CAS starts with changing R_p and R_n to $R_p^{'}$ and $R_n^{'}$ by the following conditions:

$$\left[R_p\right]_{ij}^{'} = \begin{cases} \left[R_p\right]_{ij} + random(0,1) \; if \; \left[R_p\right]_{ij} > 0 \\ 0, otherwise \end{cases} \quad \textbf{(10)}$$

$$\left[R_n\right]_{ij}^{'} = \begin{cases} \left[R_n\right]_{ij} + random(0,1) \; if \; \left[R_n\right]_{ij} > 0 \\ 0, otherwise \end{cases} \quad \textbf{(11)}$$

Example 3. The modified rule matrices from (10) and (11) shown as below

$$R^{+} = \begin{array}{c|cccccccc} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ \hline 1 & 0.00 & 0.00 & 3.09 & 5.04 & 0.00 & 0.00 & 0.00 & 0.00 \\ 2 & 0.00 & 0.00 & 0.00 & 0.00 & 1.28 & 2.31 & 4.20 & 1.98 \\ 3 & 1.73 & 0.00 & 1.33 & 1.15 & 3.50 & 1.72 & 0.00 & 1.39 \\ 4 & 1.99 & 3.18 & 1.16 & 0.00 & 1.15 & 0.00 & 1.13 & 1.84 \\ 5 & 1.62 & 1.91 & 2.13 & 1.31 & 2.02 & 0.00 & 1.96 & 0.00 \\ 6 & 3.51 & 0.00 & 1.82 & 0.00 & 3.88 & 0.00 & 1.53 & 0.00 \end{array}$$

$$R^{-} = \begin{array}{c|cccccccc} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ \hline 1 & 0.00 & 0.00 & 3.17 & 5.49 & 0.00 & 0.00 & 0.00 & 0.00 \\ 2 & 0.00 & 0.00 & 0.00 & 0.00 & 1.81 & 2.23 & 1.21 & 4.11 \\ 3 & 1.85 & 0.00 & 2.44 & 0.00 & 1.23 & 0.00 & 3.38 & 1.99 \\ 4 & 2.59 & 0.00 & 0.00 & 0.00 & 2.26 & 3.75 & 1.68 & 0.00 \\ 5 & 1.51 & 3.58 & 1.18 & 2.20 & 0.00 & 1.93 & 0.00 & 0.00 \\ 6 & 1.03 & 0.00 & 4.55 & 0.00 & 1.25 & 0.00 & 2.15 & 0.00 \end{array}$$

Theorem 1. For $\left[R_p\right]_{ij}^{'}$ and $\left[R_n\right]_{ij}^{'}$ are the elements of the matrices $R_p^{'}$ and $R_n^{'}$ which are the rule matrices that modified with equations (10) and (11) at the i^{th} row and the j^{th} column, respectively. We can denote the relation

$$A_{ij} = \left| \left[R_p\right]_{ij}^{'} - \left[R_n\right]_{ij}^{'} \right|$$

where A_{ij} is the difference of the matrices $R_p^{'}$ and $R_n^{'}$ at the same position.

If A_{ij} is exactly different, it will make the classification of patterns easier. On the other hand, if the above differences are not apparent or have a small value, the classification will be difficult.

Proof. The initial rule matrices, $R_p^{'}$ and $R_n^{'}$ can use the same element position of each other to classify the pattern.

For example, at the row 4^{th} and column 2^{nd} of $R_p^{'}$ and $R_n^{'}$, $\left[R_p\right]_{4,2}^{'}$ = 3.18 and $\left[R_n\right]_{4,2}^{'}$ = 0.00 where $\left[R_p\right]_{4,2}^{'}$ is the element that indicates the fourth position of the pattern with a configuration "001" of positive side, and $\left[R_n\right]_{4,2}^{'}$ shows the same meaning but negative side. From this information, we immediately know that the pattern with configuration "001" at position four is a positive side data since all negative data do not have any data with the configuration "001" in position forth.

On the other hand, for $\left[R_p\right]_{3,1}^{'}$ =1.73 and $\left[R_n\right]_{3,1}^{'}$ = 1.85 means the pattern with the configuration "000" in that first position. We cannot identify the side of the pattern since the values in both positions are similar.

From the above relation, we can conclude that if the A_{ij} value of any element is high, the classification ability is high. Differently, if the value of A_{ij} is low, it shows that the position also has a little classification ability.

3.2.1 Create Soft Decision Table (τ)

The first step to pruning useless branches for pruning methods is to evaluate the ability of classification in every decision tree, which can compare to the element of the rule matrices in the classifier based on Cellular automata.

Definition 1. *Overlap Coefficient*

Overlap coefficient is a measure of the ability to classify each element in the rule matrices. We use $\alpha_{ij} = \{x | x \in [0,1]\}$ to represent the *Overlap coefficient* at the i^{th} row, j^{th} column of the Rule matrices for $i=\{1,2,...,n\}$ and $j=\{1,2,...,8\}$ where n is the features number of dataset in binary patterns. From *Theorem 1* and *Definition 1*, we can define in equation (12) as below

$$\alpha_{ij} \begin{cases} \dfrac{\left|[R_p]'_{ij} - [R_n]'_{ij}\right|}{[R_p]'_{ij} + [R_n]'_{ij}}, [R_p]'_{ij} + [R_n]'_{ij} > 0 \\ 0, otherwise \end{cases} \quad \textbf{(12)}$$

where $[R_p]'_{ij}$ and $[R_n]'_{ij}$ are elements of the matrices R'_p and R'_n at the i^{th} row and the j^{th} column, respectively. α_{ij} vary with the efficiency of classification.

Definition 2. *Soft Decision Table*

Soft decision table is the relationship of α_{ij} in the form of a matrix (τ) as shown below.

$$\tau = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{18} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{28} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{n8} \end{bmatrix}$$

where n is the features number of datasets in binary patterns.

3.2.2 Adjust the Rule Matrices for BOA Processes

When the matrices R'_p, R'_n and τ will be processed by the following processes:

First, Rearrange the matrices R'_p, R'_n and to one dimension. Second, sort the one-dimensional matrices in ascending order and unique (no copy of elements allowed). Finally, the results will be contained in one dimensional matrix K_p, K_n and K_β respectively.

Definition 3. *Hard Decision Boundary*

Hard Decision Boundary $\omega_p = \{x | 0 < x < \max(K_p)\}$ and $\omega_n = \{x | 0 < x < \max(K_n)\}$ are the least value that separates the area of the classes by using an optimization algorithm to find the optimal decision boundaries for the best classification accuracy.

Definition 4. *Soft decision boundary*

The Soft decision boundary $\omega_\beta = \{x | 0 < x < \max(K_\beta)\}$ and $\omega_n = \{x | 0 < x < \max(K_n)\}$ are the best value for pruning method by using an optimization algorithm to find the optimal decision boundaries for the best classification accuracy.

3.2.2 Adjust the Rule Matrices Form for BOA

The result from butterfly optimization algorithm consists of 3 variables, $(\omega_p, \omega_n, \omega_\beta)$ which are a threshold making the classifier converge to the best solution.

3.3 Soft Decision

The pruning method is implemented on R'_p and R'_n to eliminate useless classification elements by following condition:

$$[R^+]'_{ij} = \begin{cases} [R_p]'_{ij} \; if \; [R_p]'_{ij} > [R_n]'_{ij} \; or \; \alpha_{ij} > \omega_\beta \\ 0, \; otherwise \end{cases} \quad \textbf{(13)}$$

$$[R^-]'_{ij} = \begin{cases} [R_n]'_{ij} \; if \; [R_n]'_{ij} > [R_p]'_{ij} \; or \; \alpha_{ij} > \omega_\beta \\ 0, \; otherwise \end{cases} \quad \textbf{(14)}$$

where ω_β is a threshold to making the model converged to the best answer.

3.4 Synthesizing Rule Matrices

This step is the final step in creating the rule to be used in recognition, using ω_p and ω_n from butterfly optimization process by the following condition:

$$[R^+]_{ij} = \begin{cases} 1 \; if \; [R^+]'_{ij} > \omega_p \\ 0, \; otherwise \end{cases} \quad \textbf{(15)}$$

$$[R^-]_{ij} = \begin{cases} 1 \; if \; [R^-]'_{ij} > \omega_n \\ 0, \; otherwise \end{cases} \quad \textbf{(16)}$$

where $[R^+]_{ij}$ and $[R^-]_{ij}$ are rules matrices for pattern recognition.

Although CAS uses the same process for synthesizing the rule matrices with CAC, CAS has improved an initial rule matrices format before the rule matrix synthesis. Allowing us to get a better-quality rule matrix than the previous method

Algorithm 2: CAS Algorithm

```
input : Dataset $D$
output: $R^+$, $R^-$
1  Initialization values of rule matrices;
2  if $D$ is not binary patterns then
3      Convert $Dataset$ to binary pattern by using
         Gray code.;
4      second if;
5  end
6  Create matrices $R_p$ and $R_n$;
7  Preparing data for BOA;
8  Create $R_p^{'}$ and $R_n^{'}$ by equation (10) and (11);
9  Create Soft Decision Table by equation (12);
10 Adjust matrices $K_p$,$K_n$ and $K_\beta$ for BOA process;
11 Process BOA by Algorithm 1;
12 Soft decision :;
13 Process pruning method by equation (13) and (14);
14 Sysnthesize rule matices :;
15 Synthesize rule matrices $R^+$ and $R^-$ by equation
     (15) and (16);
```

4 Experiments

In this section, we propose a process to evaluate the efficiency of the proposed classifier. The datasets, comparing classifiers and discussion are presented.

4.1 Datasets

The datasets from the UCI data repository are implemented. They consist of different types of instances, dimensions, and distribution as shown in Table 1. Datasets are analyzed through Principal component analysis (PCA). As shown PCA explicitly illustrates datasets with conforming and non-conforming patterns, we reduce a dimension of datasets to 2 dimensions by using the PCA technique and scatter plot to represent data related to each class.

Table 1. Datasets

Dataset	Class	Instances	Attributes
Monk Problem 2	2	601	7
Congressional Voting Records	2	435	16
Hayes-Roth	3	160	5
Colon	2	62	2000
Soybean	4	47	35
Pima Indians Diabetes	2	768	8
Mammographic Mass	2	961	6
New thyroid	2	306	3
SPECT heart	2	80	23
Libras Movement	15	360	91

The result shows that the datasets represent itself as clusters such as the Congressional Voting dataset (Figure 4(a)), Soybean dataset (Figure 4(c)), and Mammographic Mass dataset (Figure 4€) distribution clustered that mean datasets are conforming. But for non-conforming data, for example, The Monk's Problem 2 (Figure 4(b)), Hayes-Roth (Figure 4(d)), and SPECT heart (Figure 4(f)). The results of the scatter plot shown most of the data of different classes are mixed because the classification accuracy lowers than the previous one.

4.2 Evaluation Method

To validate the proposed method, we prepare training and sampling data by using K-fold cross validation [54] with 2, 4, 6, 8, 10 folds, respectively. According to CAS. Confusion matrix [29] is determined to report the classification accuracy as the equation given below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{17}$$

where True Positive (*TP*) is predicted to be positive correctly, False Negative (*FN*) is positive observation, but is predicted negative, True Negative (*TN*) is negative observation, and is predicted to be negative, False Positive (*FP*) is negative observation, but is predicted positive. Additionally, CAS implements DDAG scheme for multiclass classification.

4.3 Compared Methods

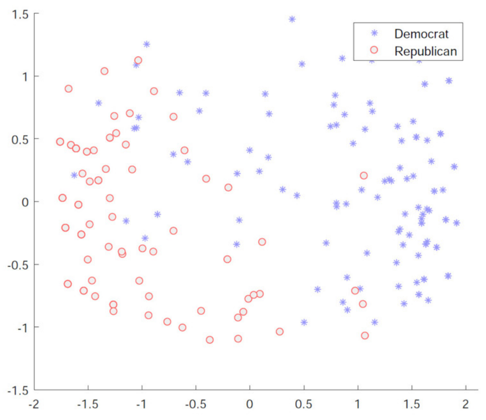
For comparison purpose, the state-of-the-art classification methods are compared. We implement the compared methods besides CAC through RapidMiner Software by setting up a test environment on a personal computer consist with 8 Gigabytes memory, CPU Intel core i7 eight cores, Windows 10 operating system. The compared methods are as follows.

4.3.1 Cellular Automata-based Classifier (CAC)

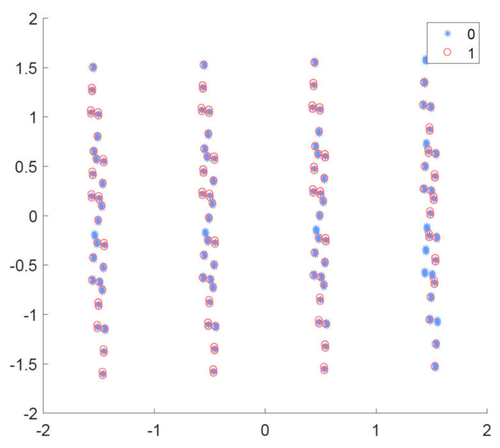
CAC is an efficient classifier based on cellular automata model that possesses the promising capability to deal with non-conforming patterns in the bit-level features [52]. It consists of 2 layers, evolving layers, and decision layer. An evolving layer consist of 2 cellular automata rule matrices to provide the next state. A decision layer processes the result of an evolving layer to decision the class of the pattern. The efficiency of the classifier is a rule matrix that created using reverse engineering techniques and genetic algorithms.

4.3.2 Support Vector Machine (SVM)

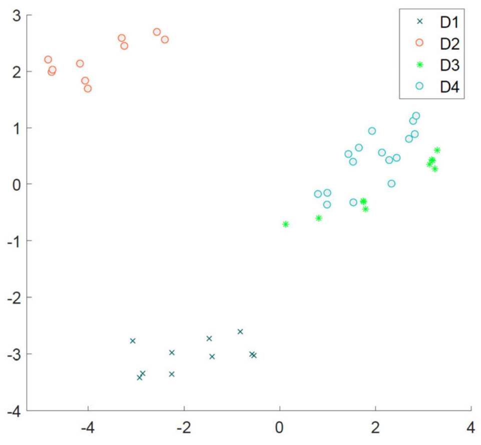
SVM is a classifier that has the principle of changing dimensions to data and finding boundaries [23], dividing data into two parts, using vectors of multiple vector points called support vector to define the boundaries. For the SVM configuration, a dot kernel type is used.



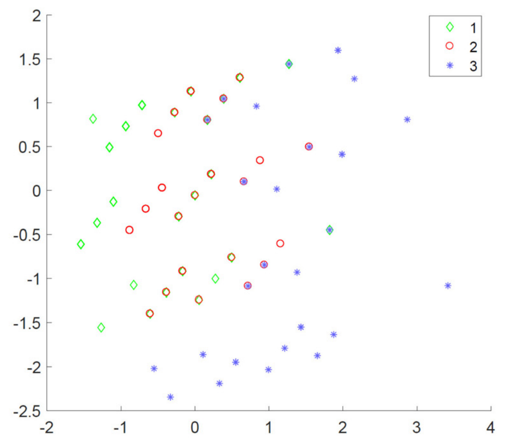
(a) Congressional voting



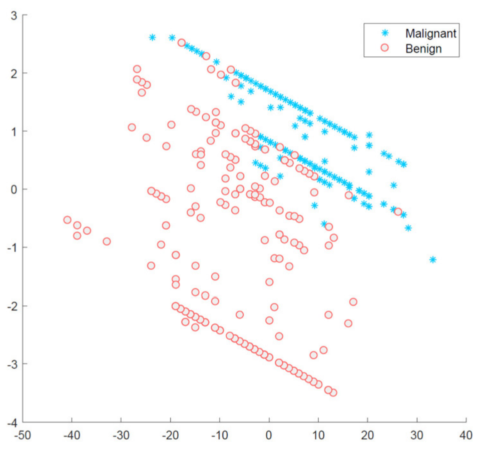
(b) The Monk's Problem 2



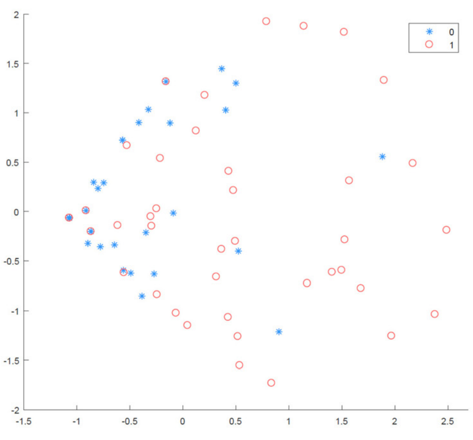
(c) Soybean



(d) Hayes-Roth



(e) Mammographic mass



(f) SPECT heart

Figure 4. Datasets visualized through PCA technique

4.3.3 k-Nearest Neighbor Algorithm (k-NN)

The k-Nearest Neighbor algorithm [9] is based on comparing an unknown example with the k training examples which are the nearest neighbors of the unknown example. In this case, we assign the number of 5 nearest neighbor to the method.

4.3.4 Deep Learning

Deep Learning is based on a multi-layer feed-forward artificial neural network that is trained with stochastic gradient descent using back-propagation [37]. The network can contain many hidden layers consisting of neurons with tanh, rectifier, and max out activation functions. We add 2 layers and 50 nodes for each layer.

4.3.5 Naive Bayes

Naive Bayes is a high-bias, low-variance classifier, and it can build a good model even with a small data set [25]. It is simple to use and computationally inexpensive. Typical use cases involve text categorization, including spam detection, sentiment analysis, and recommender systems.

4.4 Experimental Results

The recognition performance of the proposed CAS is evaluated through the stated datasets compared to the state-of-the-art methods. For non-binary features, the datasets must be transforming into gray code. For implementation purpose, the parameters of the butterfly optimization for rules ordering are set as follows:

(1) Probability switch (p): 0.5
(2) Power exponent: 0.4
(3) Sensory modality: 0.03
(4) Max iteration: 20

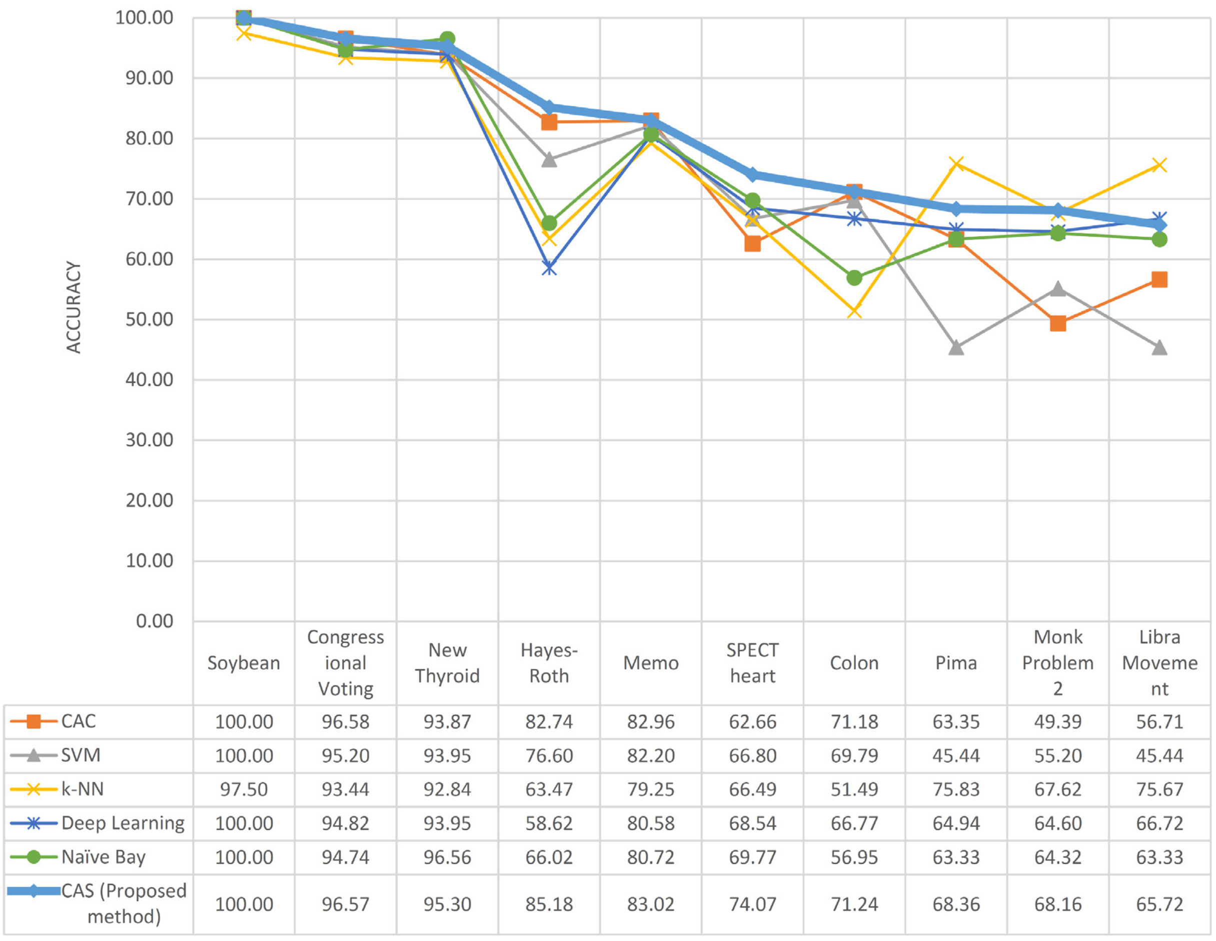
The experiments are conducted on ten UCI datasets in different instances and feature numbers. When comparing the classification accuracy result between CAC and CAS, both also report the high classification accuracy and do not seem to make much difference. Whereas the dataset with non-conforming patterns, the classification accuracy is much lower than conforming datasets. However, if we are only considering the correctness of classification between CAC and the CAS, it seems that when the datasets consist of conforming patterns, the accuracy of CAS much higher than non-conforming patterns. Table 2 shows comparative classification accuracy of the proposed algorithm and the compared state-of-the-art methods. Figure 5 portrays comparative results of classification accuracy obtained from Table 2.

Table 2. Comparative classification accuracy of the proposed algorithm and the state-of-the-art methods

Datasets	Classifiers	K-Fold					Average
		2	4	6	8	10	
Congressional Voting	CAS	96.12	96.98	96.14	97.00	96.60	96.57
	CAC	96.98	96.26	95.83	96.87	96.94	96.58
	SVM	94.00	95.00	96.00	96.00	95.00	95.20
	k-NN	93.97	93.10	93.06	93.10	93.95	93.44
	Deep Learning	95.26	93.53	94.79	95.26	95.25	94.82
	Naïve Bayes	93.97	94.83	94.8	95.26	94.82	94.74
Soybean	CAS	100.00	100.00	100.00	100.00	100.00	100.00
	CAC	100.00	100.00	100.00	100.00	100.00	100.00
	SVM	100.00	100.00	100.00	100.00	100.00	100.00
	k-NN	95.74	97.92	97.92	97.92	98.00	97.5
	Deep Learning	100.00	100.00	100.00	100.00	100.00	100.00
	Naïve Bayes	100.00	100.00	100.00	100.00	100.00	100.00
Monk Problem 2	CAS	67.05	68.55	70.21	65.97	69.02	68.16
	CAC	51.30	53.40	44.75	46.08	51.43	49.39
	SVM	55.00	58.00	54.00	56.00	53.00	55.20
	k-NN	70.05	68.06	66.89	66.56	66.56	67.62
	Deep Learning	66.56	62.40	62.23	65.93	65.89	64.60
	Naïve Bayes	63.22	64.39	65.22	64.39	64.39	64.32
Colon	CAS	72.58	74.17	73.83	69.64	65.95	71.24
	CAC	72.57	74.05	73.65	69.53	66.12	71.18
	SVM	64.52	71.15	69.70	71.21	72.38	69.79
	k-NN	54.84	60.83	50.15	43.53	48.10	51.49
	Deep Learning	66.13	69.35	66.13	64.52	67.74	66.77
	Naïve Bayes	51.61	63.02	56.36	54.24	59.52	56.95
Hayes-Roth	CAS	86.23	85.71	80.63	87.56	85.75	85.18
	CAC	81.88	80.00	84.33	85.00	82.47	82.74
	SVM	72.00	79.00	78.00	77.00	77.00	76.60
	k-NN	58.75	61.25	66.12	65.00	66.25	63.47
	Deep Learning	56.25	60.62	58.71	58.75	58.75	58.62
	Naïve Bayes	61.25	63.12	68.85	66.88	70.00	66.02
SPECT heart	CAS	76.25	73.75	72.53	75.34	72.50	74.07
	CAC	66.25	58.75	62.55	63.75	62.00	62.66
	SVM	64.00	61.00	69.00	74.00	66.00	66.80
	k-NN	71.25	67.50	66.21	63.75	63.75	66.49
	Deep Learning	68.75	66.25	68.96	68.75	70.00	68.54
	Naïve Bayes	73.75	70.00	67.58	68.75	68.75	69.77

Table 2. Comparative classification accuracy of the proposed algorithm and the state-of-the-art methods (continue)

Datasets	Classifiers	K-Fold					Average
		2	4	6	8	10	
Memo	CAS	83.53	82.25	83.15	82.53	83.65	83.02
	CAC	83.25	84.65	82.53	82.13	82.25	82.96
	SVM	82.00	83.00	82.00	82.00	82.00	82.20
	k-NN	78.80	78.92	79.76	79.39	79.40	79.25
	Deep Learning	80.00	80.60	81.20	80.48	80.60	80.58
	Naïve Bayes	79.64	80.60	81.33	81.08	80.96	80.72
Pima	CAS	72.58	74.17	73.83	69.64	65.95	71.24
	CAC	72.57	74.05	73.65	69.53	66.12	71.18
	SVM	64.52	71.15	69.70	71.21	72.38	69.79
	k-NN	54.84	60.83	50.15	43.53	48.10	51.49
	Deep Learning	66.13	69.35	66.13	64.52	67.74	66.77
	Naïve Bayes	51.61	63.02	56.36	54.24	59.52	56.95
New thyroid	CAS	94.79	94.76	96.22	95.92	94.80	95.30
	CAC	95.81	91.20	93.03	94.90	94.39	93.87
	SVM	93.96	93.96	93.98	93.93	93.92	93.95
	k-NN	91.63	92.57	93.98	92.54	93.46	92.84
	Deep Learning	93.03	94.88	94.91	93.47	93.48	93.95
	Naïve Bay	95.82	96.75	96.76	96.74	96.73	96.56
Libras Movement	CAS	62.5	65.556	64.44	65.83	70.26	65.72
	CAC	56.11	56.11	55.21	57.32	56.23	56.20
	SVM	46.94	47.78	50.28	52.22	30	45.44
	k-NN	70.83	74.44	77.5	76.67	78.89	75.67
	Deep Learning	66.67	64.72	66.11	69.44	66.67	66.72
	Naïve Bay	59.72	64.72	65.28	64.44	62.5	63.33



	Soybean	Congressional Voting	New Thyroid	Hayes-Roth	Memo	SPECT heart	Colon	Pima	Monk Problem 2	Libra Movement
CAC	100.00	96.58	93.87	82.74	82.96	62.66	71.18	63.35	49.39	56.71
SVM	100.00	95.20	93.95	76.60	82.20	66.80	69.79	45.44	55.20	45.44
k-NN	97.50	93.44	92.84	63.47	79.25	66.49	51.49	75.83	67.62	75.67
Deep Learning	100.00	94.82	93.95	58.62	80.58	68.54	66.77	64.94	64.60	66.72
Naïve Bay	100.00	94.74	96.56	66.02	80.72	69.77	56.95	63.33	64.32	63.33
CAS (Proposed method)	100.00	96.57	95.30	85.18	83.02	74.07	71.24	68.36	68.16	65.72

Figure 5. Comparative results of classification accuracy

5 Conclusion

CAC is an efficient classifier that can deal with conforming and nonconforming binary patterns. However, finding decision boundaries to divide data is difficult. In this regard, GA cannot well handle high dimensional complicated problems. In this paper, we propose an improved cellular automata-based classifier by augmenting a soft decision technique and Butterfly Optimization Algorithm in lieu of Genetic Algorithm.

As compared to the promising state-of-the-art classification methods, namely, SVM, k-NN, Deep Learning, Naïve Bayes, and CAC, CAS reports the promising results while experimented in ten UCI datasets in different instances, features, and class number.

For future research, the following issues can improve the accuracy performance. Firstly, an efficient method rather than Gray code in transforming data into binary must be determined. Secondly, multi-classification using DDAG scheme in binary classifier (CAC and CAS) is limited. An efficient technique can be determined to improve the accuracy performance.

Acknowledgments

Authors would like to thank Computer Science Department and Senior Research Scholar Program of Khon Kaen University, Thailand.

References

[1] A. Aghaei, A Cellular Automata Approach for Noisy Images Edge Detection under Null Boundary Conditions, *Second International Conference on Computing Methodologies and Communication (ICCMC)*, Erode, India, 2018, pp. 771-777.

[2] E. Al-Hawri, N. Correia, A. Barradas, Dag-coder: Directed Acyclic Graph-based Network Coding for Reliable Wireless Sensor Networks, *IEEE Access*, Vol. 8, pp. 21886-21896, January, 2020.

[3] A. Wood, V. Shpilrain, K. Najarian, D. Kahrobaei, Private Naive Bayes Classification of Personal Biomedical Data: Application in Cancer Data Analysis, *Computers in Biology and Medicine*, Vol. 105, pp. 144 -150, February, 2019.

[4] Y. T. Chang, W. K. T. M. Gunarathne, T. K. Shih, Deep Learning Approaches for Dynamic Object Understanding and Defect Detection, *Journal of Internet Technology*, Vol. 21, No. 3, pp. 783-790, May, 2020.

[5] S. Wang, H. Yi, L. Wu, F. C. Zhou, N. N. Xiong, Mining Probabilistic Representative Gathering Patterns for Mobile Sensor Data, *Journal of Internet Technology*, Vol. 18, No. 2, pp. 321-332, March, 2017.

[6] S. Arora, P. Anand, Binary Butterfly Optimization Approaches for Feature Selection, *Expert Systems with Applications*, Vol. 116, pp. 147 -160, February, 2019.

[7] S. Arora, S. Singh, Butterfly Optimization Algorithm: A Novel Approach for Global Optimization, *Soft Computing*, Vol. 23, No. 3, pp. 715-734, February, 2019.

[8] M. Bennasar, Y. Hicks, R. Setchi, Feature Selection Using Joint Mutual Information Maximisation, *Expert Systems with Applications*, Vol. 42, No. 22, pp. 8520-8532, December, 2015.

[9] B. Zhang, S. N. Srihari, Fast k-nearest Neighbor Classification Using Cluster-based Trees, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 4, pp. 525-528, April, 2004.

[10] F. Cao, W. Guo, Cascaded Dual-scale Crossover Network for Hyperspectral Image Classification, *Knowledge-Based Systems*, Vol. 189, p. 105122, February, 2020.

[11] H. Cao, S. Bernard, R. Sabourin, L. Heutte, Random Forest Dissimilarity Based Multi-view Learning for Radiomics Application, *Pattern Recognition*, Vol. 88, pp. 185-197, April, 2019.

[12] B. Chang, R. Yang, C. Guo, S. Ge, L. Li, A New Application of Optimized Random Forest Algorithms in Intelligent Fault Location of Rudders, *IEEE Access*, Vol. 7, pp. 94276-94283, July, 2019.

[13] S. Bhandari, H. P. Zhao, H. Kim, P. Khan, S. Ullah, Packet Scheduling Using SVM Models in Wireless Communication Networks, *Journal of Internet Technology*, Vol. 20, No. 5, pp. 1505-1512, September, 2019.

[14] N. R. da Silva, P. V. der Weeën, B. D. Baets, O. M. Bruno, Improved Texture Image Classification through the Use of a Corrosion-inspired Cellular Automaton, *Neurocomputing*, Vol. 149, pp. 1560- 1572, February, 2015.

[15] M. Dembowski, B. Wolnik, W. Bołt, J. M. Baetens, B. Baets, Affine Continuous Cellular Automata Solving the Fixed-length Density Classification Problem, *Natural Computing: An International Journal*, Vol. 17, No. 3, pp. 467-477, September, 2018.

[16] D. R. Don, I. E. Iacob, Dcsvm: Fast Multi-class Classification Using Support Vector Machines, *International Journal of Machine Learning and Cybernetics*, Vol. 11, No. 2, pp. 433-447, February, 2020.

[17] Fawad, M. J. Khan, M. A. Riaz, H. Shahid, M. S. Khan, Y. Amin, J. Loo, H. Tenhunen, Texture Representation through Overlapped Multi-oriented Tri-scale Local Binary Pattern, *IEEE Access*, Vol. 7, pp. 66668-66679, May, 2019.

[18] N. Gan, S. Yao, Y. Xiong, and X. Hong, A Hybrid Cellular Automaton-bi-directional Evolutionary Optimization Algorithm for Topological Optimization of Crashworthiness, *Engineering Optimization*, Vol. 50, No. 12, pp. 2054-2070, 2018.

[19] S. B. Gelfand, C. S. Ravishankar, E. J. Delp, An Iterative Growing and Pruning Algorithm for Classification Tree Design, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 2, pp. 163-174, February, 1991.

[20] J. Gou, W. Qiu, Z. Yi, X. Shen, Y. Zhan, W. Ou, Locality Constrained Representation-based k-nearest Neighbor Classification, *Knowledge-Based Systems*, Vol. 167, pp. 38-52, March, 2019.

[21] H. Gu, T. Lin, X. Wang, A Preliminary Geometric Structure Simplification for Principal Component Analysis, *Neurocomputing*, Vol. 336, pp. 46-55, April, 2019.

[22] G. Gutierres, R. Mamede, and J. L. Santos, Gray Codes for Signed Involutions, *Discrete Mathematics*, Vol. 341, No. 9, pp. 2590-2601, September, 2018.

[23] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support Vector Machines, *IEEE Intelligent Systems and Their Applications*, Vol. 13, No. 4, pp. 18-28, July-August, 1998.

[24] S. M. Hedjazi, S. S. Marjani, Pruned Genetic Algorithm, in: F. L. Wang, H. Deng, Y. Gao, J. Lei (Eds.), *Artificial Intelligence and Computational Intelligence*, Springer Berlin Heidelberg, 2010, pp. 193-200.

[25] H. J. Huang, C. N. Hsu, Bayesian Classification for Data from the Same Unknown Class, *IEEE Transactions on Systems, Man, and Cybernetics*, Part B (Cybernetics), Vol. 32, No. 2, pp. 137-145, April, 2002.

[26] A. K. Kar, Bio Inspired Computing- A Review of Algorithms and Scope of Applications, *Expert Systems with Applications*, Vol. 59, pp. 20-32, October, 2016.

[27] Z. Laboudi, An Effective Approach for Solving the Density Classification Task by Cellular Automata, *4th World Conference on Complex Systems (WCCS)*, Ouarzazate, Morocco, 2019, pp. 1-8.

[28] H. K. Lee and S. B. Kim, An Overlap-sensitive Margin Classifier for Imbalanced and Overlapping Data, *Expert Systems with Applications*, Vol. 98, pp. 72-83, May, 2018.

[29] Y. Lei, Y. Dong, F. Xiong, H. Bai, H. Yuan, Confusion Weighted Loss for Ambiguous Classification, *IEEE Visual Communications and Image Processing (VCIP)*, Taichung, Taiwan, 2018, pp. 1-4.

[30] G. Li, F. Shuang, P. Zhao, C. Le, An Improved Butterfly Optimization Algorithm for Engineering Design Problems Using the Cross-entropy Method, *Symmetry*, Vol. 11, No. 8, 1049, August, 2019.

[31] L. Li, Y. Zhang, W. Chen, S. K. Bose, M. Zukerman, G. Shen, Naïve Bayes Classifier-assisted Least Loaded Routing for Circuit-switched Networks, *IEEE Access*, Vol. 7, pp. 11854-11867, January, 2019.

[32] Y. Lin, Y. Tu, Z. Dou, An Improved Neural Network Pruning Technology for Automatic Modulation Classification in Edge Devices, *IEEE Transactions on Vehicular Technology*, Vol. 69, No. 5, pp. 5703-5706, May, 2020.

[33] Y. Liu, H. H. Zhang, Y. Wu, Hard or Soft Classification? Large-margin Unified Machines, *Journal of the American Statistical Association*, Vol. 106, No. 493, pp. 166-177, March, 2011.

[34] P. Maji, P. P. Chaudhuri, Non-uniform Cellular Automata Based Associative Memory: Evolutionary Design and Basins of Attraction, *Information Sciences*, Vol. 178, No. 10, pp. 2315-2336, May, 2008.

[35] E. N. Mambou, T. G. Swart, A Construction for Balancing Non-binary Sequences Based on Gray Code Prefixes, *IEEE Transactions on Information Theory*, Vol. 64, No. 8, pp. 5961-5969, August, 2018.

[36] Y. Pan, Z. Pan, Y. Wang, W. Wang, A New Fast Search Algorithm for Exact k-nearest Neighbors Based on Optimal Triangle-inequality-based Check Strategy, *Knowledge-Based Systems*, Vol. 189, p. 105088, February, 2020.

[37] M. Zhou, Z. Bai, T. Yi, X. Chen, W. Wei, Performance Predict Method Based on Neural Architecture Search, *Journal of Internet Technology*, Vol. 21, No. 2, pp. 385-392, March, 2020.

[38] H. M. Proença, M. van Leeuwen, Interpretable Multiclass Classification by Mdl-based Rule Lists, *Information Sciences*, Vol. 512, pp. 1372-1393, February, 2020.

[39] L. C. Ribas, J. Machicao, O. M. Bruno, Life-like network Automata Descriptor Based on Binary Patterns for Network Classification, *Information Sciences*, Vol. 515, pp. 156-168, April, 2020.

[40] A. Sasithradevi, S. M. M. Roomi, Video Classification and Retrieval Through Spatio-temporal Radon Features, *Pattern Recognition*, Vol. 99, p. 107099, March, 2020.

[41] S. Dehuri, A. Ghosh, Revisiting Evolutionary Algorithms in Feature Selection and Nonfuzzy/Fuzzy Rule Based Classification, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 3, No. 2, pp. 83-108, March/April, 2013.

[42] S. Sharma, A. K. Saha, m-mboa: A Novel Butterfly Optimization Algorithm Enhanced with Mutualism Scheme, *Soft Computing*, Vol. 24, No. 7, pp. 4809-4827, April, 2020.

[43] C. Sompong, S. Wongthanavasu, An Efficient Brain Tumor Segmentation Based on Cellular Automata and Improved Tumor-cut Algorithm, *Expert Systems with Applications*, Vol. 72, pp. 231-244, April, 2017.

[44] J. Song, P. Shen, K. Wang, L. Zhang, H. Song, Can Gray Code Improve the Performance of Distributed Video Coding? *IEEE Access*, Vol. 4, pp. 4431-4441, August, 2016.

[45] Y. Song, F. Wang, X. Chen, An Improved Genetic Algorithm for Numerical Function Optimization, *Applied Intelligence*, Vol. 49, No. 5, p. 1880-1902, May, 2019.

[46] P. Songsiri, T. Phetkaew, B. Kijsirikul, Enhancement of Multi-class Support Vector Machine Construction from Binary Learners Using Generalization Performance, *Neurocomputing*, Vol. 151, pp. 434-448, March, 2015.

[47] M. Tang, R. Pérez-Fernández, B. D. Baets, Fusing Absolute and Relative Information for Augmenting the Method of Nearest Neighbors for Ordinal Classification, *Information Fusion*, Vol. 56, pp. 128-140, April, 2020.

[48] G. E. Tucker, D. E. J. Hobley, E. Hutton, N. M. Gasparini, E. Istanbulluoglu, J. M. Adams, S. S. Nudurupati, Celllab-CTS 2015: Continuous-time Stochastic Cellular Automaton Modeling Using Landlab, *Geoscientific Model Development*, Vol. 9, No. 2, pp. 823-839, February, 2016.

[49] Y. Sung, J. Kwak, J. H. Park, Decision Tree Generation Algorithm for Image-based Video Conferencing, *Journal of Internet Technology*, Vol. 20, No. 5, pp. 1535-1545, September, 2019.

[50] S. Wang, P. Lin, R. Hu, H. Wang, J. He, Q. Huang, S. Chang, Acceleration of lstm with Structured Pruning Method on fpga, *IEEE Access*, Vol. 7, pp. 62930-62937, May, 2019.

[51] Z. Wang, Z. Zhu, D. Li, Collaborative and Geometric Multi-

kernel Learning for Multi-class Classification, *Pattern Recognition*, Vol. 99, p. 107050, March, 2020.

[52] P. Wanna, S. Wongthanavasu, J. Ponkaew, A Differential Evolution-based Rule Ordering of Cellular Automata for Classification, *10th International Conference on Knowledge and Smart Technology (KST)*, Chiang Mai, Thailand, 2018, pp. 34-39.

[53] B. Wolnik, M. Dembowski, W. Bołt, J. M. Baetens, B. D. Baets, Density-conserving Affine Continuous Cellular Automata Solving the Relaxed Density Classification Problem, *Journal of Physics A: Mathematical and Theoretical*, Vol. 50, No. 34, p. 345103, July, 2017.

[54] T. Wong, N. Yang, Dependency Analysis of Accuracy Estimates in k-fold cross Validation, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 29, No. 11, pp. 2417-2427, November, 2017.

[55] S. Wongthanavasu, J. Ponkaew, A Cellular Automata-based Learning Method for Classification, *Expert Systems with Applications*, Vol. 49, pp. 99-111, May, 2016.

[56] Z. Yilbas, M. Hashmi, Simulation of Weight Prunning Process in Backpropagation Neural Network for Pattern Classification: A Self-running threshold Approach, *Computer Methods in Applied Mechanics and Engineering*, Vol. 166, No. 3-4, pp. 233-246, November, 1998.

[57] B. Zhang, X. Yang, B. Hu, Z. Liu, Z. Li, Oebboa: A Novel Improved Binary Butterfly Optimization Approaches with Various Strategies for Feature Selection, *IEEE Access*, Vol. 8, pp. 67799-67812, April, 2020.

[58] T. Zhang, J. Liang, B. Ding, Acoustic Scene Classification Using Deep CNN with Fine-resolution Feature, *Expert Systems with Applications*, Vol. 143, p. 113067, April, 2020.

[59] Y. Zhao, K. Hao, H. He, X. Tang, B. Wei, A Visual Long-short-term Memory Based Integrated CNN MODel for Fabric Defect Image Classification, *Neurocomputing*, Vol. 380, pp. 259-270, March, 2020.

[60] H. S. Zhao, Y. Gao, H. Liu, L. Li, Fault Diagnosis of Wind Turbine Bearing Based on Stochastic Subspace Identification and Multi-kernel Support Vector Machine, *Journal of Modern Power Systems and Clean Energy*, Vol. 7, No. 2, pp. 350-356, March, 2019.

Biographies

Pattapon Wanna received B.S. in Electrical Engineering from Burapha University, Thailand, in 2006 and M.Sc. in Information Technology from King Mongkut’s University of Technology Thonburi (KMUTT). His research interests include Machine Learning, Cellular Automata. He is a Ph.D. cancidate at Computer Science Department, Khon Kaen University, Thailand.

Sartra Wongthanavasu received M.S. in computer science from Illinois Institute of Technology, USA, in 1996. He received Ph.D. in computer science from Asian Institute of Technology, Thailand, in 2001. His research interests include machine learning, computer vision, cellular automata, knowledge engineering.