Journal of Engineering Science and Technology
Vol. 12, No. 6 (2017) 1446 - 1459

AN ADABOOST OPTIMIZED CCFIS BASED CLASSIFICATION MODEL FOR BREAST CANCER DETECTION

CHANDRASEKAR RAVI, NEELU KHARE*

School of Information Technology and Engineering (SITE), VIT University, Vellore Campus, Vellore-632014, Tamil Nadu, India
*Corresponding Author: neelu.khare@vit.ac.in

Abstract

Classification is a Data Mining technique used for building a prototype of the data behaviour, using which an unseen data can be classified into one of the defined classes. Several researchers have proposed classification techniques but most of them did not emphasis much on the misclassified instances and storage space. In this paper, a classification model is proposed that takes into account the misclassified instances and storage space. The classification model is efficiently developed using a tree structure for reducing the storage complexity and uses single scan of the dataset. During the training phase, Class-based Closed Frequent ItemSets (CCFIS) were mined from the training dataset in the form of a tree structure. The classification model has been developed using the CCFIS and a similarity measure based on Longest Common Subsequence (LCS). Further, the Particle Swarm Optimization algorithm is applied on the generated CCFIS, which assigns weights to the itemsets and their associated classes. Most of the classifiers are correctly classifying the common instances but they misclassify the rare instances. In view of that, AdaBoost algorithm has been used to boost the weights of the misclassified instances in the previous round so as to include them in the training phase to classify the rare instances. This improves the accuracy of the classification model. During the testing phase, the classification model is used to classify the instances of the test dataset. Breast Cancer dataset from UCI repository is used for experiment. Experimental analysis shows that the accuracy of the proposed classification model outperforms the PSO-AdaBoost-Sequence classifier by 7% superior to other approaches like Naïve Bayes Classifier, Support Vector Machine Classifier, Instance Based Classifier, ID3 Classifier, J48 Classifier, etc.

Keywords: Association mining, Class-based closed frequent itemset, Tree structure, Ensemble classifier, Optimization.

Nomenclatures

a, b, Wt	Temporary variables
$A=\{A_1, A_2, \ldots, A_m\}$	The set of attributes
a,b,Wt	Temporary variables
$A=\{A_1, A_2, \ldots, A_m\}$	The set of attributes
accuracy	Accuracy of the classifier
$alpha_k$	Importance of the k^{th} classifier
Attribute	Attributes of an itemset
$C=\{C_1, C_2, \ldots, C_p\}$	The set of class labels
c_1 and c_2	Cognitive & social learning factor
$CCFIS_i$	Class-based Closed Frequent Itemset of the i^{th} class
class	Class into which an instance has been classified
Confidence	Confidence of an itemset
correct	No. of correctly classified instances
CTREE	Class-based closed frequent itemset tree
CW	Weights of the classes
D	Sum of no. of sample training itemsets and no. of class
Db	Dataset
error	Error of the classifier
Itemset	Itemset stored in the node
l	No. of sample training itemsets
Level	Level of *CTREE*
m	Number of attributes
minConfidence	Confidence threshold
minSupport	Support threshold
n	Number of records
N	Number of rounds of boosting
Node	Node of *CTREE*
p	Number of class
Pn	Population size
$R=\{R_1, R_2, .., R_n\}$	The set of records
$Recordset_i$	Set of record IDs that contains the itemset and class C_i
$STIS_k$	Similarity value
TestD	Sample training itemsets in k^{th} round of booosting
total	Test set
TrainD	Sum of all $\|Recordset_i\|$ of an itemset
Values	Training set
V_{max}	Values corresponding to the attribute in an itemset
$Weights_k$	Velocity restriction of the particle
W_k	Weights if the itemset during the k^{th} round of boosting
wrong	Wt. of the sample training itemsets in k^{th} round of boosting No. of wrongly classified instances

Abbreviations

CAR	Class Association Rule
CCFIS	Class-based Closed Frequent Itemset Mining
CAR	Class Association Rule

CCFIS	Class-based Closed Frequent Itemset Mining
CFIS	Class-based Frequent Itemset Mining
FP	Frequent Pattern
LCS	Longest Common Subsequence
PSO	Particle SwarmOptimization

1. Introduction

Data Mining has grasped the attention of the society in the recent past. This is mainly because of the wide amount of data availability and the immense need for generating information and knowledge from the data. Rich information is hidden in the data that can be used for decision making purpose. Classification is a Data Mining technique that analyses the data and extracts model depicting the data classes. Rules are a good form of representing information. So, Rule based classification has gained more popularity. It consists of two steps, namely, rule generation and classification. Association Rule Mining is one of the most common technique used for generating the classification rules. A variety of classifiers have been proposed in the literature for classification.

Agarwal et al. proposed Association Rule Mining [1] for mining the rules and Apriori [2] for generating frequent itemsets. The limitations of Apriori are that it produces many candidate itemsets and scans the dataset multiple times. Since then, several algorithms [3 - 8] were proposed to reduce the mining time. These algorithms did not emphasis on the class to which the itemset belongs to. Hence the class-based frequent itemset mining algorithms were researched in the past. These techniques can be broadly categorized into three groups, namely, Generate-and-Test techniques, Divide-and-Conquer techniques and Lattice-based techniques. The Generate-and Test techniques [9 - 12] suffered from multiple scans of the datasets as they mostly inherited from Apriori. The Divide-and-Conquer techniques [7, 13, 14] used efficient data structures derived from Frequent Pattern Tree (FP-Tree). But FP-Tree does not give importance to class-based itemsets which is the main drawback of these techniques. Lattice-based techniques [15 - 17] also suffered from multiple scan of the dataset.

Algorithms for mining Class Association Rules [18 - 22] mainly suffered from enormous amount of rules generated. Although algorithms for mining Class Association Rules [23 - 30] were proposed to reduce the number of rules, they were not successful in efficient mining of interesting rules. They did not focus enough on the size and efficiency of the storage structure. Vo et al. [22], proposed a tree structure for mining frequent itemsets using Equivalence Class. It scans the dataset only once, but it is time consuming because the itemsets belonging to same attribute were placed in the same node of the tree. Loan et al. [31] modified the tree structure [22] to include one itemset per node of the tree. Dang et al. [32] optimised the tree [31] to generate only the nodes which follows the class constraints. This reduced the computational complexity. Closed frequent itemsets are compact representation of the dataset rather than the frequent itemsets. In view of the above, this paper, the tree structure [32] has been enhanced to contain only the nodes with closed itemsets thereby optimising the tree and reducing the space and time complexity of generating CCFIS. This enhanced tree, named as C-Tree, is used in this work to generate the class-based closed frequent itemsets which is the input to the classifier model.

Algorithms [33 - 36], proposed for classification, are based on Association Rules. But they do not signify the effect of the misclassified instances. Chieh et al. [37] proposed a classifier based on a partial similarity measure called Longest Common Subsequence in which itemsets and the classes were assigned weights optimally generated by the Particle Swarm Optimization (PSO) Algorithm. The weightage of the misclassified instances are dynamically adapted in each round of boosting to classify the rare instances. This technique was used for classification of sequence data.

The above technique has been extended in this paper to classify the transaction data using the proposed CCFIS stored in tree structure. An ensemble classifier has been proposed based on Class-based Association Rule mining algorithms and a similarity approach based on Longest Common Subsequence (LCS). CCFIS are derived based on Class Association Rules (CARs) and fed to an ensemble similarity based classifier which is optimized using Particle Swarm Optimization (PSO) algorithm. There are three main contributions in this work. Firstly, the tree [32] has been enhanced and the C-Tree has been proposed for mining and storing the CCFIS efficiently. Secondly, the classification system [37] has been enhanced for classifying transactional data rather than sequence data, which emphasises misclassified instances. Finally, breast cancer dataset from UCI machine learning repository has been used for experimentation and the accuracy of the proposed methodology has been evaluated and compared with that of benchmark classification system. The rest of the paper is organized in the following manner: the theory behind the proposed methodology, the experimental results, discussions and the conclusion.

2. Proposed Method

The architecture of the proposed classification system, shown in Fig. 1, consists of two phases, namely, the training phase and the testing phase. The dataset is partitioned into training dataset and testing dataset. During the training phase, CCFIS are mined into a tree structure with a single scan of the training set. The Particle Swarm Optimization algorithm is used to determine weights to the class-based closed frequent itemsets. Then, training dataset and the weighted CCFIS samples are fed into the similarity measurement based classifier. Further, The AdaBoost algorithm is applied to boost accuracy of the classifier by repeating the classification process iteratively. During each iteration, the weights of the misclassified instances are adaptively increased. At the end of the training phase, several sets of weighted CCFIS samples are obtained by the classifier at the rate of per set per round of boosting and the importance for each iteration of boosting is determined. During the testing phase, the test dataset is fed as input to the classification model. In order to classify each instance of the test dataset, the classifier consolidates the decision obtained during each round on the basis of the importance given to each round of boosting. Finally, each test instance is labelled with a class.

Let *Db* be the dataset. Let $R=\{R_1,R_2,..,R_n\}$ be the set of instances or records of the dataset *Db* where *n* is the number of records. Let $A=\{A_1,A_2,\ldots,A_m\}$ be the set of attributes where *m* is the number of attributes. Let $C=\{C_1,C_2,\ldots,C_p\}$ be the set of class labels where *p* is the number of classes. Let the attribute-value pair, i.e. (A_k,a_{xk}) represent an item, where $k \in [1, m]$, and its corresponding value a_{xk} in the

x^{th} records, where $k \in [1, m]$ and $x \in [1, n]$. Let $\{(A_j,a_{xj}) ,...,(A_k,a_{xk})\}$ be an itemset (set of items), where $j, k \in [1, m]$, $j \neq k$ and $x \in [1, n]$. Let $\{(A_j,a_{xj}) ,...,(A_k,a_{xk})\} \rightarrow C_i$ be a class-based itemset belonging to class C_i, where $i \in [1, p]$. Let *Supp* represent the support count that is the number of records in *Db* that matches the itemset and its associated class label.

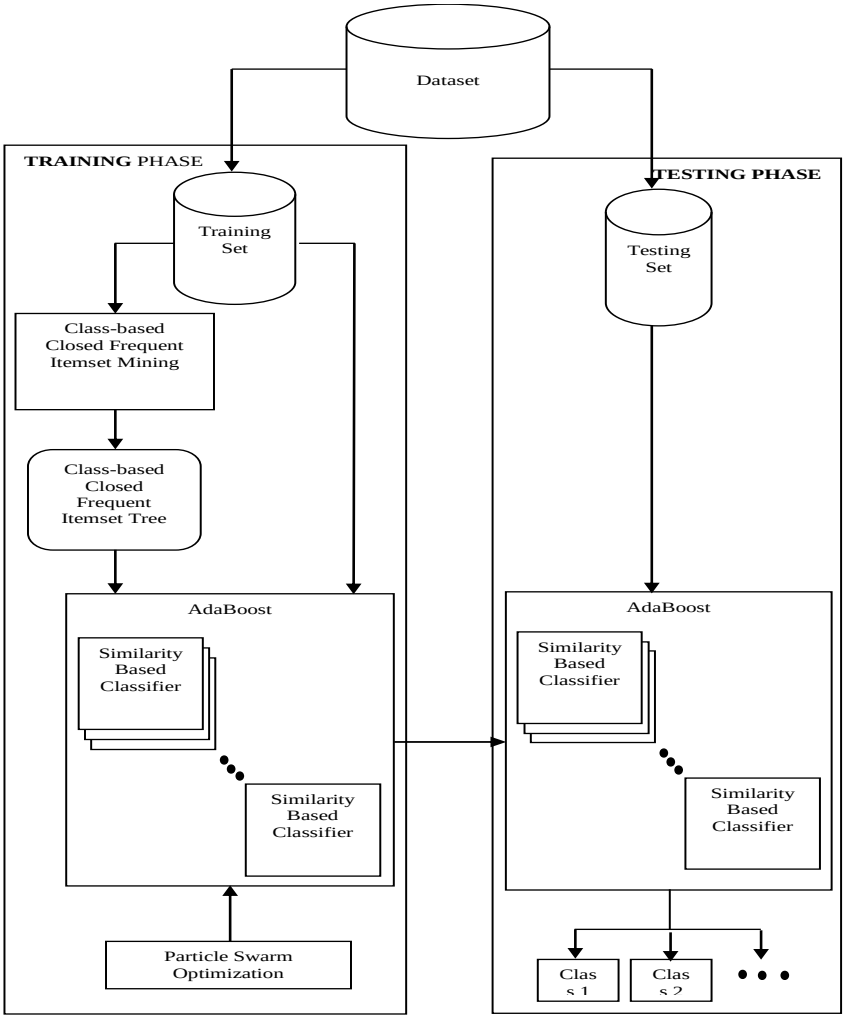


Fig. 1. Architecture of the proposed classification system.

An itemset can belong to any number of classes. But only the class in which the itemset has the maximum support count is considered as the class for the itemset while the others are ignored. If the support count is the same and maximum for more than a class then the itemset will be considered in all the classes. This assigns a maximum support count for every class-based itemset. Let *minSupp* be the support count threshold. An itemset is said to be frequent if *Supp* > *minSupp.* An itemset is said to be closed frequent if it is frequent and none of its immediate supersets has the same support count as the itemset has.

Dang et al. [32] proposed a tree which stores the frequent itemsets by a single scan of the dataset. They proposed an efficient algorithm for mining CARs using the tree. Frequent itemsets may be too many and enumerating all of them might be costly and ineffective. On the other hand, Closed frequent itemsets are sufficient to represent the itemsets. Hence, in this work, the tree [32] is enhanced for mining CCFIS, rather than the frequent itemsets, and name it as C-Tree. Moreover, the nodes in the proposed tree structure stores only 4-tuple information rather than 5-tuple information, as in the case of the previous tree, thereby eliminating the redundant information. The structure of the C-Tree is explained below. Each node of the C-Tree represents a frequent itemset and stores the following information, namely, *Attribute, Value, {Recordset*$_1$*, Recordset*$_{2,\ldots,}$ *Recordset*$_p$*}* and *Class* where $|Node.Recordset_{Class}| > (|Node.Recordset_i|_{i\in[1,p]})$. *Attribute* stores the set of attributes that the itemset represent. *Value* stores the values corresponding to the set of attributes represented by *Attribute*. Each $Recordset_i$ where $i \in [1, p]$, represents the set of record IDs that contains the itemset and corresponds to class C_i.

The proposed algorithm for mining CCFIS, named CCFIS Miner, is shown in Fig. 2. The training dataset is scanned once and the frequent 1-itemsets are extracted and added to the C-Tree. CFIS_Miner is used to build the C-Tree. Line 8 takes each node of the C-Tree and compares it with the other nodes in the same level of tree. This is repeated till all the nodes in the C-Tree are traversed. In Line 9, two nodes are considered if their attributes are different. The *Attribute*, *Value*, *Recordset* and *Class* of the new itemset is derived in Line 10-13. *Class* is the class of the itemset for which the itemset has the maximum support count. Line 14 checks if the itemset is frequent. If so, the new itemset is inserted as the child of the parent itemset in Line 15. When all the new itemsets in this level are complete, the CFIS_Miner algorithm is called recursively until all the itemsets are mined. The CCFIS_Miner algorithm builds the set of CCFIS which is given as input for training the classifier.

Tsai et al. [37] proposed a sequence classifier using a similarity measure. The accuracy of the classifier was boosted using Adaboost and Particle Swarm Optimization. In their work, the classifier consumed the compact sequential patterns extracted using CloSpan algorithm from Sequence dataset. In this work, their algorithms are enhanced to train and test the classifier using the CCFIS generated by the CCFIS_Miner, rather than the compact sequential patterns extracted using CloSpan algorithm from Sequence dataset. The proposed algorithm for the AdaBoost classification system is shown in Fig. 3. Line 3 of the algorithm chooses a sample of CCFIS into *STIS* for each round *k* based on the weights *W* of the CCFISs. The weights *W* for the initial round is equal to 1/*m* for all the samples in line 1, where *m* is the number of samples. Not all the itemsets in $CCFIS_i$, where i ∈ [1, *p*], are equally important.

Similarly, not all classes are equally important. The importance of various classes and the itemsets in each class are decided by the weight assigned for each in $Weights_k$ during each of the *k* rounds. This $Weights_k$ for each round *k* is determined by the PSO algorithm in line 4. Line 5-12 estimates the *error* of the classifier for each of the *k* rounds of boosting using the $Weights_k$. Again, each classifier is not equally important. The importance *alpha* of each of the *k* classifiers is determined by the *error* of the classifier in line 18. Lines 19-23 update the weights of the misclassified itemsets in *STIS*. Thus, at the end of *N*

rounds, *N* classifiers are built. Lines 25-36 validates the classifier models using the test dataset *TestD* and the *accuracy* is calculated in line 37. The proposed similarity-based classifier is shown in Fig. 4.

```
Input          : Training Dataset TrainD, minSupport
Output         : Class-based Closed Frequent ItemSet pertaining to class i,
                 where i ∈ [1, p] and satisfying minSupport.
Procedure CFIS_Miner
1. CTree_root = Ø
2. CCFIS_i = Ø, for all i ∈ [1, p]
3. CTree_1 = {frequent 1-itemset}
4. level = 1
5. repeat
6.      Node = Ø
7.      for all itemset_i ∈ CTree_level
8.          for all itemset_j ∈ CTree_level and j > i
9.              if itemset_i.Attribute ≠ itemset_j.Attribute
10.                 Node.Attribute = itemset_i.Attribute U
                    itemset_j.Attribute
11.                 Node.Values = itemset_i.Values U
                    itemset_j.Values
12.                 Node.Recordset = itemset_i.Recordset ∩
                    itemset_j.Recordset
13.                 Node.Class=k,where |Node.Recordset_k| >
                    (|Node.Recordset_i|_i∈[1,p])
14.                 if |Node.Recordset_Class| ⩾ minSupport
15.                     Insert Node as the child of
                        itemset_i
16.                 end
17.             end
18.         end
19.     end
20. level++
21. while (Node ≠ Ø)
22. CCFIS_Miner (CTree)
end

Sub-Procedure CCFIS_Miner (CTree)
23. for each Node ∈ CTree
24.     Class=k, where |Node.Recordset_k| > (|Node.Recordset_i|_i∈[1,p])
25.     if |Node.Recordset_Class| ≠ |Node_child.Recordset_Class|
26.         Insert the itemset of Node in CCFIS_Class
27.     end
28. end
```

Fig. 2. The proposed algorithm for mining class-based closed frequent ItemSet.

Input : Class-based closed frequent itemset, $CCFIS_i$, where i ∈ [1, p]
No. of rounds of boosting N, Training Dataset *TrainD* , Testing Dataset *TestD*, Particle population *pn*, Velocity restriction of the particles V_{max}, the cognition learning factor c_1 and social learning factor c_2.
Output : The *accuracy* of the classifier

Procedure AdaBoost

1. $W_1(1{:}l) = 1/l$
2. for $k = 1$ to N
3. Create the Sample Training ItemSet $STIS_k$ by sampling *CCFIS* based on W_k
4. $Weights_k$=PSO ($STIS_k$, *TrainD*, *pn*, V_{max}, c_1, c_2)
5. for $i = 1$ to l
6. if Classifier($STIS_k$, $STIS_{ki}$, $Weights_k$) ≠ C_i
7. $flag = 1$
8. else
9. $flag = 0$
10. end
11. $error = error + (flag * W_k(i))$
12. end
13. $error = error / l$
14. if $error \geq 0.5$
15. $W(1{:}\ l) = 1/l$
16. GOTO step 3
17. end
18. $alpha_k = 0.5 * \ln((1\text{-}error)/error)$
19. if $flag = 0$
20. $W_{k+1}(i) = W_k(i) * \exp \wedge (\text{-}alpha_k)$
21. else
22. $W_{k+1}(i) = W_k(i) * \exp \wedge alpha_k$
23. end
24. end
25. for $i = 1$ to |*TestD*|
26. for $j = 1$ to N
27. C = Classifier($STIS_j$, $TestD_i$, $Weights_k$)
28. $flag_C = flag_C + alpha_j$
29. end
30. *class* = indexof(max(*flag*))
31. if $class = C_i$
32. *correct*++
33. else
34. *wrong*++
35. end
36. end
37. return $accuracy = correct$ / |TestD|

end

Fig. 3. The proposed algorithm for the AdaBoost classification system.

Input : *STIS*, *Instance*, *Weights*
Output : *class* of the instance
Procedure Classifier
1. Extract the weights of the *STIS* belonging to class c in W_c
2. Extract the weights of the class in *CW*
3. for $c = 1$ to p
4. for $i = 1$ to $|TIS_c|$
5. $Wt(c) = Wt(c) + W_c(i)$ * Similarity ($STIS_c(i)$, *Instance*)
6. end
7. $Wt(c) = (Wt(c) / |TIS_c|)$ * $CW(c)$
8. end
9. Return *class* = indexof(max(*Wt*))

Sub-Procedure Similarity(a, b)
10. $S = |(a \cap b)|/(\max(|a|,|b|))$
end

Fig. 4. The proposed algorithm for the similarity-based classifier.

The instance that is to be classified, is compared with all the weighted itemsets in each class separately. The classifier calculates the similarity of the instance with all the itemsets in each class. The weightage of the various classes are also considered. The class of *STIS* having the highest similarity with the instance to be classified is considered as the target class of the instance. The similarity estimation is done from line 3 to 8. The target class is determined in step 9. The proposed PSO algorithm for optimizing the weights of the itemsets in each round is shown in Fig. 5. Line 2 initializes the particles in the swarm. Lines 8-10 identifies the local maximum and lines 11-13 identifies the global maxima. Lines 15-18 update the weight and velocity of the particles for the next round. Lines 6-19 repeat this until any stopping criteria is reached. Line 20 returns the particle with the highest fitness value. Lines 21-29 in the sub-procedure calculates the accuracy of classification. All these are repeated for *N* rounds. At the end of *N rounds*, the training phase is over and the training model is built. The proposed AdaBoost algorithm, shown in Fig. 3, classifies the testing set *TestD* from lines 25-36 and the *accuracy* is calculated in line 37.

3. Experimental Results and Discussions

The experiments were performed in a system with Intel Core i5 - 4200U CPU @ 1.60GHz 2.30GHz, 8GB RAM and 64-bit Windows 8.1 Operating System. The breast-cancer-wisconsin dataset from UCI machine learning repository (https://archive.ics.uci.edu/ml/datasets.html) was used for experimental purpose.

The dataset consists of 699 instances of 10 attributes namely Sample code number, Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell, Bare Nuclei, Bland Chromatin, Normal Nucleoli and Mitoses. All the attributes, except the Sample code number, has continuous values ranging from 1-10. All the instances were labelled as either benign or malignant. The dataset was distributed with 458 benign (65.5%) and 241 malignant (34.5%) instances. The dataset was discretized using Weka 3.6.

The proposed algorithms have been implemented using Matlab. Randomly, 70% of the dataset was used for training and the remaining 30% of the dataset was used for testing. This setup was repeated for 10 times. Each time, the training dataset was given as input to the CFIS_Miner algorithm and the CCFIS corresponding to each class were derived. This CCFIS was fed to the AdaBoost algorithm which derived a classifier model by repeating for 10 times. The PSO algorithm was executed for 50 rounds with 30 population during each round.

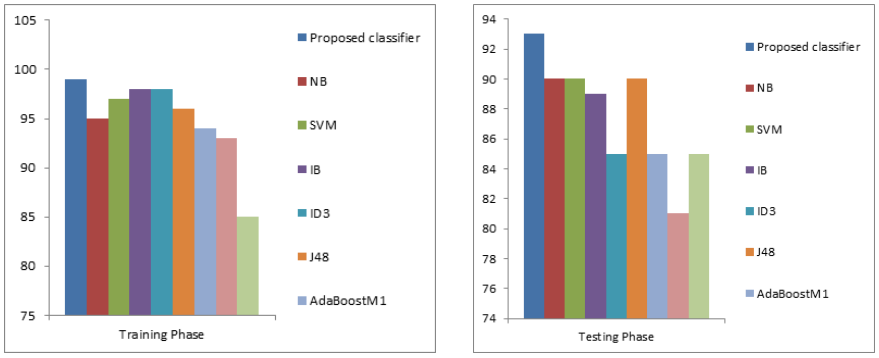
Input : *STIS*, Training Dataset *TrainD* , Particle population *pn*, Velocity restriction of the particles V_{max}, the cognition learning factor c_1 and social learning factor c_2.
Output : The particle with highest fitness value
Procedure PSO
1. $D = \sum_{c=1}^{p} |STIS\ c| + p$
2. Set $x_{i,d}^0$ as 1 for i = 1 to *pn*, d = 1 to D
3. Set $v_{i,d}^0$ as a random value in [-v_{max}, v_{max}] for i = 1 to *pn*, d = 1 to D
4. Calculate the fitness function f(x_i^0) for i = 1 to *pn*
5. t = 1
6. while (stopping criteria is not reached)
7. for i = 1 to *pn*
8. if f(x_i^t) ≥ f(x_i^{t-1})
9. $p_{id}^t = x_{id}^t$
10. end
11. if f(x_i^t) = max$_{i = 1 \text{ to } pn}$ f(x_i^t)
12. $p_{gd}^t = x_{id}^t$
13. end
14. end
15. for i = 1 to *pn*
16. $v_{id}^t = w\, v_{id}^{t-1} + c_1 r_1 (p_{id}^t - x_{id}^t) + c_2 r_2 (p_{gd}^t - x_{id}^t)$
17. $x_{id}^t = x_{id}^{t-1} + v_{id}^{t-1}$
18. end
19. end
20. return the particle with highest fitness value
end
Sub-Procedure f(x)
21. for i = 1 to |*TrainD*|
22. *class* = Classifier(*STIS*, $TrainD_i$, x)
23. if *class* = C_i
24. *correct*++
25. else
26. *wrong*++
27. end
28. end
29. return *Accuracy* = *correct* / |*TrainD*|
end

Fig. 5. The proposed particle swarm optimization algorithm.

The experimental results are tabulated in Table 1. The dataset was then tested using benchmark classifiers like Naïve Bayes, Support Vector Machine, Instance Based Classifier, ID3, J48 and other classifiers using Weka 3.6. The PSO-Sequence-Classifier [37] has been implemented using Matlab and the Malware API sequence dataset was used for evaluating the accuracy of the system. The proposed algorithm produced an accuracy of 99% when validated with training set and 93% when validated with testing set. It is observed from Fig. 6. that the accuracy of the proposed algorithm during training phase as well as the testing phase is better than that of all the benchmark and previous classifiers.

Table 1. Accuracy of the various classifiers during training phase and testing phase.

Classifier	Accuracy (%)	
	Testing Phase	Training Phase
Proposed Classifier	93	99
NB	90	95
SVM	90	97
IB	89	99
AdaboostM1	85	94
MultiBoostAB	81	93
ID3	85	99
J48	90	96
PSO-Seq-Classifier [37]	85	85



(a) Training phase. **(b) Testing Phase.**

Fig. 6. Accuracy of the various classifiers during training phase and testing phase.

4. Conclusions

In this paper, a classifier is proposed that takes into account the misclassified instances. In addition, the classification model is efficiently built using a single scan of the dataset and is stored in a tree structure for efficient storage and quicker access. During the training phase, Class-based Closed Frequent ItemSets (CCFIS) were mined from the training dataset into a tree structure. The classification model was built using the CCFIS and a similarity measure based on Longest Common Subsequence (LCS). Particle Swarm Optimization algorithm optimizes the weights used by the itemsets and classes. AdaBoost algorithm boosts the weight of the misclassified instances of the previous round so as to adapt them and include them

in the classification model to classify the atypical instances. This improves the accuracy of the classification model. During the testing phase, the classification model is used to classify the instances of the testing dataset. Breast Cancer dataset from UCI repository was used for experimental purposes. Experimental investigation shows that the accuracy of the proposed classifier is better than that of some of the benchmark classifiers like Naïve Bayes Classifier, Support Vector Machine Classifier, Instance Based Classifier, ID3 Classifier, J48 Classifier and few other Classifiers.

References

1. Agrawal, R.; Imielinski, T.; and Swami, A. (1993). Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, SIGMOD '93, 207-216.
2. Agrawal, R.; and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB* '94, 487-499.
3. Han, J.; Pei, J.; and Yin, Y. (2000). Mining frequent patterns without candidate generation. *Proceedings of 2000 ACM SIGMOD international conference on Management of data, SIGMOD* '00, 1-12.
4. Zaki, M.J.; and Hsiao, C.-J. (2005). Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering*, 17(4), 462-478.
5. Zaki, M.J.; Parthasarathy, S.; Ogihara, M.; and Li, W. (1997). New algorithms for fast discovery of association rules. *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, KDD* '97, 283-286.
6. Vo, B.; Hong, T.P.; and Le, B. (2012). DBV-Miner: a dynamic bit-vector approach for fast mining frequent closed itemsets. *Expert Systems with Application*, 39(8), 7196-7206.
7. Dong, J.; and Han, M. (2007). BitTableFI: an efficient mining frequent itemsets algorithm. *Knowledge.-Based Systems*, 20(4), 329-335.
8. Song, W.; Yang, B.; and Xu, Z. (2008). Index-BitTableFI: an improved algorithm for mining frequent itemsets. *Knowledge-Based Systems*, 21(6), 507-513.
9. Baralis, E.; Cagliero, L.; Cerquitelli, T.; and Garza, P. (2012). Generalized association rule mining with constraints. *Information Science*, 194, 68-84.
10. Cagliero, L.; and Garza, P. (2013). Itemset generalization with cardinality-based constraints. *Information Science*, 224, 161-174.
11. Raymond, T.Ng.; Lakshmanan, L.V.S.; Han, J.; and Pang, A. (1998). Exploratory mining and pruning optimizations of constrained associations rules. *Proceedings of the* 1998 *ACM SIGMOD International Conference on Management of Data, SIGMOD* '98, 13-24.
12. Srikant, R.; Vu, Q.; and Agrawal, R. (1997). Mining association rules with item constraints. *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, KDD* '97, 67-73.

13. Pei, J.; Han, J.; and Lakshmanan, L.V.S. (2001). Mining frequent itemsets with convertible constraints. *Proceedings of the 17th International Conference on Data Engineering*, 433-442.
14. Leung, C.K.S.; Lakshmanan, L.V.S.; and Raymond, T.Ng. (2002). Exploiting succinct constraints using FP-trees. *ACM SIGKDD Exploration Newsletters*, 4(1), 40-49.
15. Lu, N.; Zhou, C.G.; and Zhou, J.Z. (2005). Research on association rules mining algorithm with item constraints. *Proceedings of the* 2005 *International Conference on Cyberworlds*, *CW* '05, 329-333.
16. Tran, A.; Duong, H.; Truong, T.; and Le, B. (2011). Efficient algorithms for mining frequent itemsets with constraint. *Proceedings of the 3rd International Conference on Knowledge and Systems Engineering*, *KSE* '11, 19-25.
17. Duong, H.; Truong, T.; and Le, B. (2013). An Efficient Algorithm for Mining Frequent Itemsets with Single Constraint. *Advanced Computational Methods for Knowledge Engineering, Studies in Computational* Intelligence, 479, 367-378.
18. Cagliero, L.; and Garza, P. (2013). Improving classification models with taxonomy information. *Data Knowledge Engineering*, 86, 85-101.
19. Li, W.; Cao, L.; Zhao, D.; Cui, X.; and Yang, J. (2013). CRNN: integrating classification rules into neural network. *Proceedings of the* 2013 *International Joint Conference on Neural Networks*, *IJCNN* '13, 1-8.
20. Chen, F.; Wang, Y.; Li, M.; Wu, H.; and Tian, J. (2014). Principal association mining: an efficient classification approach. *Knowledge Based Systems*, 67, 16-25.
21. Deng, H.; Runger, G.; Tuv, E.; and Bannister, W. (2014). CBC: an associative classifier with a small number of rules. *Decision Support Systems*, 59, 163-170.
22. Vo, B.; and Le, B. (2009). A Novel Classification Algorithm Based on Association Rules Mining. *Knowledge Acquisition: Approaches, Algorithms and Applications*, 5465,61-75.
23. Toivonen, H.; Klemettinen, M.; Ronkainen, P.; Kimmo; Hätönen; and Mannila, H. (1995). Pruning and grouping discovered association rules. *Proceedings of the 1995Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*, *ECML* '95, 47-52.
24. Liu, B.; Hsu, W.; and Ma, Y. (1998). Integrating classification and association rule mining. *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, *KDD* '98, 80-86.
25. Nguyen, L.T.T.; Vo, B.; Hong, T.P.; and Thanh, H.C. (2012). Classification based on association rules: a lattice-based approach. *Expert Systems Applications*, 39(13), 11357-11366.
26. Liu, H.; Liu, L.; and Zhang, H. (2011). A fast pruning redundant rule method using Galois connection. *Applications Soft Computing*, 11(1), 130-137.
27. Li, J.; and Cercone, N. (2005). Discovering and ranking important rules. *Proceedings of the 2005 IEEE International Conference on Granular Computing*, 506-511.

28. Najeeb, M.; Sheikh, A.E.; and Nababteh, M. (2011). A new rule ranking model for associative classification using a hybrid artificial intelligence technique. *Proceedings of the 3rd International Conference on Communication Software and Networks, ICCSN* '11, 231-235.
29. Cai, R.; Tung, A.; Zhang, Z.; and Hao, Z. (2011). What is unequal among the equals? Ranking equivalent rules from gene expression data. *IEEE Transactions on Knowledge Data Engineering*, 23(11), 1735-1747.
30. Chen, C.H.; Chiang, R.D.; Lee, C.M.; and Chen, C.-Y. (2012). Improving the performance of association classifiers by rule prioritization. *Knowledge Based Systems*, 36, 59-67.
31. Nguyen, L.T.T.; Vo, B.; Hong, T.P.; and Thanh, H.C. (2013). CAR-Miner: an efficient algorithm for mining class-association rules. *Expert System. Applications*, 40(6), 2305-2311.
32. Nguyen, D.; Nguyen, L.T.T.; Vo, B.; and Hong, T.P. (2015). A novel method for constrained class association rule mining. *Information Sciences*, 320, 107-125.
33. Liu, B.; Hsu, W.; and Ma, Y. (1998). Integrating classification and association rule mining. *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, KDD* '98, 80-86.
34. Li, W.; Han, J.; and Pei, J. (2001). CMAR: accurate and efficient classification based on multiple class-association rules. *Proceedings of the 2001 IEEE International Conference on Data Mining, ICDM* '01, 369-376.
35. Yin, X.; and Han, J. (2003). CPAR: classification based on predictive association rules. *Proceedings of the 3rd SIAM International Conference on Data Mining, SDM* '03, 331-335.
36. Thabtah, F.; Cowling, P.; and Peng, Y. (2004). MMAC: a new multi-class, multi-label associative classification approach. *Proceedings of the 4th IEEE International Conference on Data Mining, ICDM* '04, 217-224.
37. Tsai, C.Y.; and Chen, C.J. (2015). A PSO-AB classifier for solving sequence classification problems. *Applied Soft Computing*, 27, 11-27.