Recursive Harmony Search Based Classifier Ensemble Reduction

Prajwal Kailas
Department of Computer Science and Engineering
National Institute of Technology
Karnataka Surathkal, India
prajwal967@gmail.com

K Chandrasekaran
Department of Computer Science and Engineering
National Institute of Technology
Karnataka Surathkal, India
kch@nitk.ac.in

***Abstract*—In recent times classifier ensembles have become a mainstay in data mining and machine learning. The combination of several classifiers generally results in better performance and accuracy as compared to a single classifier. There are many different methods and techniques for constructing ensembles. Most of the time however, when these ensemble classifiers are constructed, the data used in the construction of ensemble classifiers becomes redundant. This redundant data results in a loss of accuracy and an increase in memory and system overhead. Therefore by removing this redundant data we can reduce the memory and system overhead as well as obtain an increase in accuracy. The redundant data can be eliminated by using a technique called feature selection. Feature selection is used to select the most relevant features while performing any task. There are many different feature selection algorithms such as memetic algorithms, sub-modular feature selection, etc. The feature selection technique can be used to choose the relevant data and eliminate the redundant data. The way to eliminate redundant data in ensemble classifiers is to perform classifier ensemble reduction. This paper discusses using feature selection and in particular employing recursive harmony search to perform classifier ensemble reduction via feature selection. The final ensemble classifier will be a reduced set of the original ensemble classifier, while maintaining diversity and accuracy of the original one.**

Keywords*—*Machine Learning; Ensemble Classifier; Feature Selection; Harmony Search; Recursive Harmony Search

I. INTRODUCTION

Ensemble learning is a machine learning approach where multiple classifiers are trained to solve the same problem. Ensemble classifiers construct a set of hypothesis and combine them as opposed to learning from the training data using a single hypothesis [1].

The main objective of ensemble classifiers is to improve the performance of stand alone classifiers. Each classifier has its own solution space derived from a hypothesis, by combining different models, the different hypotheses are combined to produce a more robust solution space, which in turn increases the accuracy as compared to having just a single solution space and hypothesis. Essentially there is a combination of the various diversities of single classifiers to obtain a more diverse model. There are many techniques used while building classifier ensembles. Some of them involve training the single classifier on different subsets of the data, this increases the accuracy as well as reduces the complexity of training single classifiers with big datasets. The mainstream approach to build an ensemble classifier involves 2 essential steps. The first step involves selecting a group of classifiers, ideally with diverse training methods and hypothesis. The second step involves selecting an ensemble approach to apply on this group of classifiers. Some of the approaches are bagging, boosting, stacking, blending, etc.

In most cases the ensemble classifier has a higher accuracy as compared to a single classifier. In some cases when the group of classifiers is redundant, that is when one or more classifiers yield the same prediction and diversity, there is no increase in accuracy and may also lead to a decrease in accuracy. Another clear problem is that the ensemble classifier which is a group of classifiers will take more time to train and test as it involves training, testing and obtaining predictions from every single classifier in the group. Therefore the time and data constraints result in a significant overhead to memory and run time.

The objective of classifier ensemble reduction [2] is to overcome the drawbacks of ensemble classifiers. As the name suggests we reduce the number of classifiers in the ensemble classifier; thus, by reducing the size of the ensemble classifier, the time and data requirements will reduce and therefore will reduce the system overhead. The advantages of classifier ensemble reduction are an increase in efficiency, making the process of constructing an ensemble classifier faster and smoother, reducing memory and storage requirement which is a direct result of reducing the number of classifiers, improving the diversity and performance of the ensemble classifier by eliminating redundancy within the group of classifiers.

Feature selection is used to select an optimal subset of features based on some heuristic to remove redundancy and maintain diversity [6] [7]. The main objective of feature selection is to identify a reduced feature subset for a given problem such that the data and diversity being captured remains the same, and thereby ensuring that the accuracy of predictions remain equal to or increases as compared to the accuracy of predictions on the original set of features [8] [9]. One obvious disadvantage of applying feature selection is the presence of a high number of features, that is, when the dimensionality of the data is very large [10] [11].

A standard approach to locating an optimal feature subset is performing an exhaustive search. This is however impractical for large datasets. There is another approach called hill climbing, which is to add or remove features one at time and this

is carried out as long as the performance keeps increasing and stops when no further improvement in performance is obtained. The hill climbing approach converges very fast but leads to the selection of sub optimal feature subsets [22], both in terms of the size of the subset, and performance. To overcome these drawbacks, random search algorithms and nature based heuristic algorithms were developed. Some examples are simulated annealing [23], particle swarm optimization [24] and genetic algorithms.

In this paper, a new framework for performing classifier ensemble reduction has been presented. It involves using the idea of a meta-heuristic feature selection technique called harmony search, and applying a recursive technique to get a more robust classifier ensemble reduction framework. The ensemble classifier contains a group of classifiers, each classifier in the group is considered as a feature. The objective is to select an optimal subset of features which is equivalent to selecting an optimal subset of classifiers in the ensemble classifier. The modified harmony search algorithm needs to ensure there is no loss in diversity, accuracy, performance and remove the redundancy of the individual classifiers. The implementation involves automating the process of classifier ensemble reduction, which involves automating the training and testing of the base classifiers, followed by the training and testing of the ensemble classifier, and applying the recursive harmony search based feature selection algorithm.

The combination of classifier ensemble reduction and feature selection algorithms have become an important facet in machine learning tasks requiring high accuracy and shorter run time, especially for big data. Depending on the trade off between accuracy and run time, the classifier ensemble reduction is applied. The classifier ensemble reduction also reduces the complexity of the ensemble classifier as the redundant information is removed and more explicit patterns are identified. Therefore, identifying and evaluation of different techniques of feature selection to perform classifier ensemble reduction has become an important aspect of machine learning and data mining.

The paper is organized as follows. Section II Literature Survey; Section III presents the harmony search optimization algorithm; Section IV describes the harmony search feature selection algorithm; Section V presents the recursive harmony search feature selection algorithm; Section VI Results and Analysis; Section VII Conclusion.

II. Literature Survey

Ensemble learning is a technique which uses multiple classifiers to solve the same problem. The reasons why an ensemble classifier performs better than a single classifier are as follows: the information contained in the training data might not be sufficient for a single standalone classifier. The different single classifiers will have different search spaces and methods, by combining the classifiers the imperfections in the search methods of one classifiers is compensated by another classifier. The hypothesis space of a single classifier might not represent the true target value, by using an ensemble classifier a good approximation of this value is obtained. Ensemble learning is being used in a wide range of applications such as, facial, text and optical character recognition, medical diagnosis and gene analysis [1].

Assume a dataset containing n features, that is, the dimensionality of the data is n. There are 2^n candidate subsets. The task of feature selection is to select the most optimal of these 2^n subsets. The most optimal solution may change according to the problem definition. There have been many approaches to perform feature selection in the literature. These approaches are based on probabilistic consistency [12], correlation analysis [13], fuzzy rough sets [14] [9] [15] and rough sets[16] [17]. There has also been an unsupervised feature selection approach which is performed on unlabeled data [18]. The algorithms mentioned are independent of the learning classifier being used and they are referred to as filter based approaches. There are feature selection algorithms that are dependent on the learning classifiers and are used in conjunction with learning classifiers. These feature selection techniques are hybrid algorithms and wrapper based algorithms [19] [20] [21].

Many ensemble feature selection techniques incorporate diversity as a metric in order to locate the most optimal subset of classifiers. There are a number of ways to quantify the diversity of an ensemble classifier. A fitness function is used to measure the diversity of the ensemble classifier. This fitness function captures the total ensemble diversity as well as the contribution of each individual classifier. Diversity is based on certain characteristics, such as, difference between average base classifier accuracy and ensemble accuracy, or the difference between max base classifier accuracy and ensemble classifier accuracy. Four strategies are discussed, genetic algorithm approach, hill climbing approach, ensemble backward and forward sequential selection approaches. The fitness function is integrated with the search strategies using these methods, dynamic voting and selection, weighted voting, and static selection [7].

The drawbacks of existing methods, specifically the numerical methods have brought about the advent of meta-heuristic algorithms to solve optimization problems. Meta-heuristic algorithms make use of randomness and rules to mimic natural processes. The harmony search algorithm is a meta-heuristic approach that can be used for engineering optimization problems. Harmony search uses a stochastic random search as opposed to a gradient search. The stochastic search is based on two factors, the harmony memory considering rate and pitch adjustment rate. Harmony search can be used for a wide range of applications as it imposes fewer mathematical rules [26].

The harmony search optimization algorithm has been applied to various domains by manipulating the basic anatomy of the algorithm. The algorithm was inspired from jazz musicians. The harmony search algorithm uses a novel stochastic derivative, which is based on the musicians producing the best music. The structure of the algorithm allows it to be used in problems involving discrete variables. Some of the real world applications of harmony search optimization are music composition, logistics and tour planning. In the field of computer science, harmony search has been used in web page clustering, internet planning, text summarisation, visual tracking and power system design [25].

III. Harmony Search

Harmony search is a meta heuristic algorithm which is based on finding a solution vector to optimize the cost function

of a given optimization problem. In terms of music, harmony search can be described as follows:

- Each variable or feature is considered as a musician.
- Each variable or feature generates a value which is considered as a note.
- These values are used to find the global optimum, which is the best harmony.

Essentially, harmony search is selecting the set of notes (set of features) that produces the best harmony (global optimum). Harmony search converts the feature selection problem into an optimization problem and finds the global optimum which translates to finding the optimal set of features.

The key concepts of harmony search are musicians, notes, harmonies and harmony memory [27]. To put it into perspective, the musicians are the features which become the decision variables to optimize the cost function. The feature selection problem is becoming an optimization problem. Each harmony is made up of notes and these notes are played by the musician. The harmony memory holds a selection of harmonies, and is represented by a two dimensional matrix. The rows make up the harmonies, while the columns represent the notes played by the musicians. Each column represents the note played by one musician. The harmony memory size is the maximum number of rows.

The musician is the decision variable, the musician decides whether a note should be included in the harmony or not. The harmony is composed of notes that the musicians decide on. Some musicians may choose not to include any note in the harmony. The notes selected by the musicians to form the harmony forms the feature subset. The harmony matrix will consist of all the decisions made by the musicians for each harmony/row. The next step is to evaluate the feature subset to identify the best harmony that is produced by the decisions of the musicians. This is how harmony search is translated to a feature selection problem. Harmony search feature selection is based on five parameters, harmony memory size, the number of iterations, the number of feature selectors, pitch adjustment rate and the harmony memory considering rate.

IV. Harmony Search Feature Selection Framework

The feature selection algorithm to perform classifier ensemble reduction is inspired from the harmony search algorithm. The feature selection algorithm captures the key ideas of the harmony search algorithm and models it to perform classifier ensemble reduction. The idea is that the musician will select whether to use a base model or not, that is, whether the base model should be included in the group of classifiers forming the ensemble classifier. The notes are the base classifiers and the harmony is the ensemble classifier.

A random initial harmony matrix is initialized, the initialized matrix will consist of 0s and 1s. 0 indicates that the musician has not included the base classifier in the ensemble classifier and 1 indicates that the base classifier is included in the ensemble classifier. The size of the matrix is harmony memory size x no of base classifiers. A harmony memory considering rate, pitch adjusting rate and number of iterations are also initialized. The objective is to find the optimal subset of base classifiers to form the ensemble classifier such that the diversity and accuracy are maintained. The initial matrix would look like as follows:

Initial Harmony Matrix

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

There are 10 base models to choose from, hence, the harmony search matrix has 10 columns, while there are 5 rows because we have assigned the value 5 to the harmony memory size. For example, in the first row, the first, second, fifth, sixth, and seventh base classifier will be used (1) in the ensemble classifier while the other base classifiers will not be used (0) in the ensemble classifier. Since this is an initial random matrix, this does not represent the most optimal solution. To move towards the optimal solution we employ the harmony search algorithm.

The first step of the algorithm is to identify the worst harmony in the matrix. The worst harmony is the ensemble classifier with the least accuracy. Once this row is identified we update the values of the columns of that row. This is done by using certain random variables. For each column we use a random variable and check if the random variable is greater than the harmony memory considering rate, if it is, the value of the column is updated by selecting a value randomly from any other row of that column, then we evaluate if the random variable is greater than the pitch adjustment rate, if that is true then we update the value again, if the value is a 0, it is updated to 1, and if the value is 1 it is updated to a 0. If the initial condition was false, that is, if the random value was less than the harmony memory considering rate, we randomly choose a 0 or 1 and update the value of the column. This process is done for every column of the worst harmony row. At the end of this step, a new matrix is obtained. If a row of all 0's is obtained after the update, that row is eliminated. The harmony memory considering rate and the pitch adjustment rate are values that range from [0,1]. Then, we repeat the process again, that is, we find the worst harmony, which is the ensemble classifier with the worst accuracy and update that row. This process is carried out until we exhaust the number of iterations. At the end of all the iterations the matrix would have transformed into something like this:

Final Harmony Matrix

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

To obtain the optimal feature subset, which is the optimal subset of base classifier we choose the ensemble classifier that yields the highest accuracy in the final matrix. This subset of base classifiers is the group of classifiers for constructing the ensemble classifier. The optimal subset for this example is:

Optimal Harmony Matrix

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

V. Recursive Harmony Search Feature Selection

The harmony search feature selection relies on the initial harmony search matrix to converge to an optimal solution. There may be cases where the initial matrix may lead the harmony search feature selection to converge to a local optimum rather than the global optimum. To overcome this drawback we propose an extension to the harmony search feature selection algorithm. The recursive harmony search feature selection algorithm reduces the reliance of the harmony search feature selection algorithm on the initial matrix. Recursive harmony search feature selection initializes multiple harmony matrices. Each matrix is evaluated using the harmony search feature selection algorithm and an optimal matrix is obtained from each initial randomly initialized matrix. The optimal matrices are then combined into a matrix (initial recursive harmony matrix) and the harmony search feature selection algorithm is then applied to this optimal matrix to obtain a final optimal solution. This approach reduces the reliance of the algorithm on the initial matrix, as multiple matrices followed by their solution are evaluated using the harmony search feature selection algorithm recursively. This ensures the most optimal solution for the classifier ensemble reduction. The matrices and their transformations are shown below.

Recursive Harmony Search:

- Number of Matrices: 3
- Number of Iterations: 10
- Number of Base Classifiers: 8
- Harmony Memory Size: 5
- Step 1: Initialize matrices and perform harmony search on each individual matrix.

Initial Harmony Matrix 1

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Initial Harmony Matrix 2

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Initial Harmony Matrix 3

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

- Step 2: The final matrices that are obtained after performing harmony search on the matrices in step 1 will be evaluated.

Final Harmony Matrix 1 (A row of 0's is eliminated)

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Final Harmony Matrix 2

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Final Harmony Matrix 3

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

- Step 3: Identify the optimal matrices for each individual matrix obtained from step 2.

Optimal Harmony Matrix 1

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Optimal Harmony Matrix 2

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Optimal Harmony Matrix 3

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

- Step 4: Combine the optimal matrices into a new matrix and perform harmony search on the new matrix to obtain the final harmony matrix, which is then evaluated to obtain the final optimal matrix.

Initial Recursive Harmony Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Final Recursive Harmony Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Final Recursive Optimal Harmony Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The final optimal matrix obtained at the end of step 4 is the subset of classifiers that will be used in the ensemble classifiers.

VI. Results and Analysis

The base classifiers used were a combination of decision trees, random forests, linear regression and logistic regression. The stacking ensemble classifier was used, and the algorithm used for the stacking ensemble was the logistic regression classifier. The harmony memory considering rate, pitch adjustment rate, number of iterations and the harmony memory size were initialized. Then, based on the number of base classifiers and the harmony memory size, a number of random matrices were initialized. Each randomly initialized matrix was evaluated by the harmony search feature selection algorithm.

The feature selection algorithm was applied to perform classifier ensemble reduction. The ensemble classifier was evaluated in terms of the receiver operating characteristic - area under the curve metric. The ensemble classifier, or the row with the lowest accuracy was identified and updated based

on the harmony search feature selection algorithm. At the end of the process, an optimal matrix was obtained for each randomly initialized matrix. The optimal matrices for each randomly initialized matrix were combined into a matrix and the harmony search feature selection algorithm was applied on this matrix to obtain the final optimal matrix.

The recursive harmony search feature selection algorithm was tested against 4 datasets: the bank dataset, the ionosphere dataset, the magic dataset and the heart dataset. For each dataset, we tested the performance with and without applying the recursive harmony search feature selection algorithm. There are two tables of performance for each dataset, the first table shows the performance without applying classifier ensemble reduction and the second shows the performance after classifier ensemble reduction. For example (bank dataset), the first table shows that for an ensemble classifier with all 10 models, the accuracy is 0.9422 and CPU Run-time is 1.8s, the second table shows that on applying classifier ensemble reduction, the number of models reduces from 10 to 8 (subset of 10) and the accuracy of training with 8 models is 0.9432 and the run time decreases to 1.64 seconds. 50 reduces to 25, and 100 reduces to 48 on applying the recursive harmony search feature selection algorithm (classifier ensemble reduction).

Implementation Details:

- Base Classifiers: Decision Trees, Logistic Regression, Linear Regression, Random Forest
- Ensemble Classifier: Stacking (Logistic Regression)
- Evaluation Metric: roc_auc_score
- Number of Initial Matrices: 10
- Harmony Memory Size: 5
- Harmony Memory Considering Rate: 0.7
- Pitch Adjustment Rate: 0.4

Bank Dataset:

- 41189 Instances
- 21 Features

TABLE I: Ensemble Classifiers (Bank)

Number of Models	Accuracy	CPU Time
10	0.9422	1.80s
50	0.9411	3.81s
100	0.9404	5.96

TABLE II: Classifier Ensemble Reduction (Bank)

Number of Models	Accuracy	CPU Time
8	0.9432	1.64s
25	0.9417	2.52s
48	0.9406	3.77s

Magic Dataset:

- 19021 Instances
- 11 Features

TABLE III: Ensemble Classifiers (Magic)

Number of Models	Accuracy	CPU Time
10	0.9221	1.10s
50	0.9331	1.52s
100	0.9343	1.88s

TABLE IV: Classifier Ensemble Reduction (Magic)

Number of Models	Accuracy	CPU Time
7	0.9225	0.92s
27	0.9327	1.29s
53	0.9338	1.59s

Heart Dataset:

- 270 Instances
- 14 Features

TABLE V: Ensemble Classifiers (Heart)

Number of Models	Accuracy	CPU Time
10	0.8777	0.575s
50	0.8746	0.652s
100	0.8629	0.760s

TABLE VI: Classifier Ensemble Reduction (Heart)

Number of Models	Accuracy	CPU Time
4	0.8814	0.560s
25	0.8820	0.620s
46	0.8858	0.684s

Ionosphere Dataset:

- 351 Instances
- 35 Features

TABLE VII: Ensemble Classifiers (Ionosphere)

Number of Models	Accuracy	CPU Time
10	0.9562	0.592s
50	0.9690	0.648s
100	0.9628	0.763s

TABLE VIII: Classifier Ensemble Reduction (Ionosphere)

Number of Models	Accuracy	CPU Time
4	0.9702	0.576s
19	0.9671	0.608s
56	0.9605	0.694s

The results show the performance of a reduced ensemble classifier against the performance of the original ensemble classifier. In all cases, the reduced ensemble classifier requires lesser memory and has a faster CPU run time, thereby reducing system overhead. The accuracy of the reduced ensemble classifier is either greater than, or almost equal to that of the original classifier. A visual description of the results is shown in the graphs below.

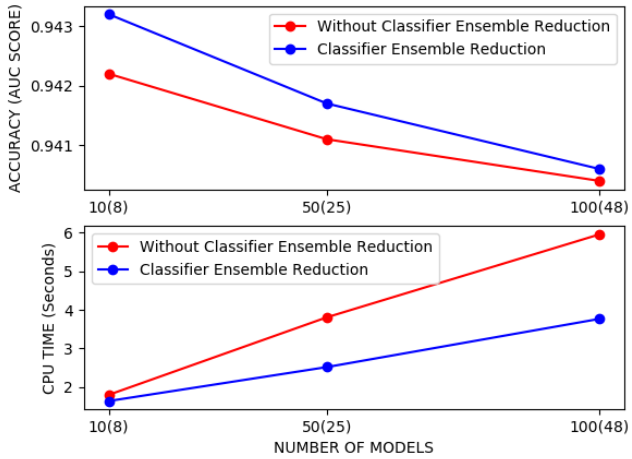

Fig. 1: Classifier Ensemble Reduction (Bank Dataset): Measuring the change in accuracy and CPU Run time before and after applying classifier ensemble reduction

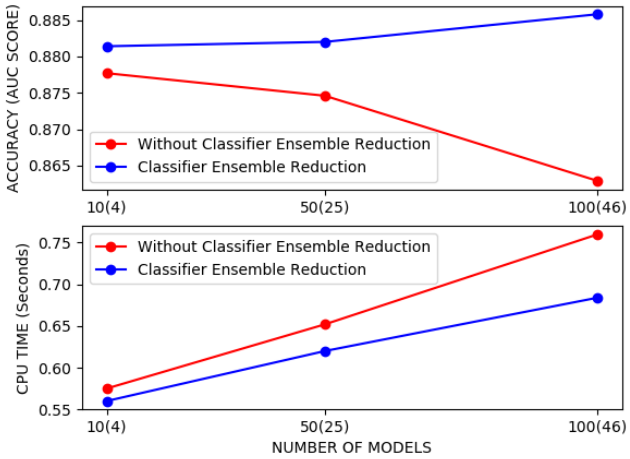

Fig. 3: Classifier Ensemble Reduction (Heart Dataset): Measuring the change in accuracy and CPU Run time before and after applying classifier ensemble reduction

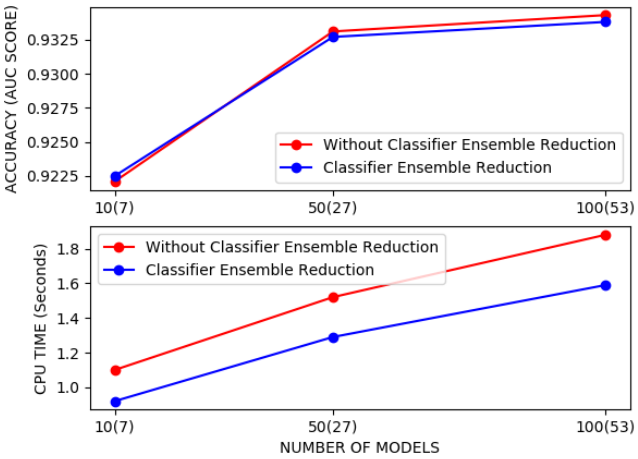

Fig. 2: Classifier Ensemble Reduction (Magic Dataset): Measuring the change in accuracy and CPU Run time before and after applying classifier ensemble reduction

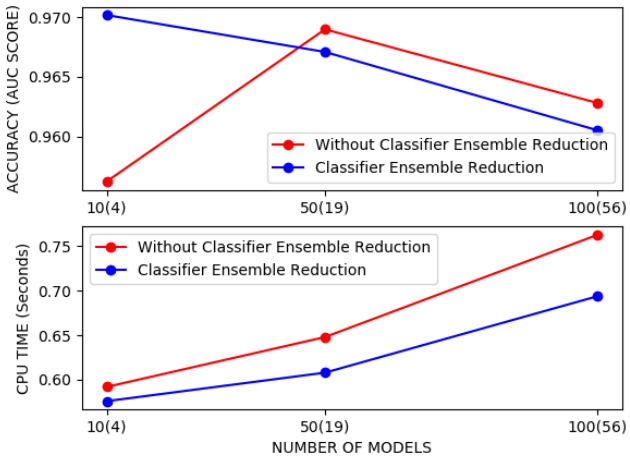

Fig. 4: Classifier Ensemble Reduction (Ionosphere Dataset): Measuring the change in accuracy and CPU Run time before and after applying classifier ensemble reduction

VII. Conclusion

This paper has presented a new robust approach for performing classifier ensemble reduction. The approach is based on a applying an optimization algorithm, harmony search, towards a feature selection problem. The harmony search feature selection technique minimizes redundancy, maintains diversity and accuracy, and reduces memory and run time requirements. The recursive harmony search feature selection ensures there is little influence of the initial matrix used in the harmony search feature selection algorithm, by recursively eliminating sub-optimal solutions. The classifier ensemble reduction results in a reduced subset of classifiers that are used to train the ensemble classifier. This reduced ensemble classifier has the accuracy and diversity similar to that of the original ensemble classifier. This reduced set of the ensemble classifier eliminates redundancy and is shown to run faster than the original set of ensemble classifier during run time/test time. This is of huge benefit to applications which require faster run times while maintaining accuracy. The feature selection algorithms remain an active area of research and much can be done to improve the algorithm presented in the paper. The reduction procedure might be performed in conjunction with decent variety upgrading strategies, for example, ensemble selection [28], [29], making the final solution differing and in addition minimized. It would be advantageous to have a closer examination in the fundamental thoughts of such comparable work, with an end goal to fabricate reduced frameworks of enhanced speculation and interpretability.

REFERENCES

[1] Zhou, Zhi-Hua., *Ensemble learning*, 2015: 411-416

[2] G. Tsoumakas, I. Partalas, and I. Vlahavas, *A taxonomy and short review of ensemble selection*, Proc. Workshop Supervised Unsupervised Ensemble Meth. Their Applicat., 2008, pp. 4146.

[3] G. Giacinto and F. Roli, *An approach to the automatic design of multiple classifier systems*, Pattern Recognit. Lett., vol. 22, no. 1, pp.2533, 2001.

[4] I. Partalas, G. Tsoumakas, and I. Vlahavas, *Pruning an ensemble of classifiers via reinforcement learning*, Neurocomputing, vol. 72, nos.79, pp. 19001909, 2009..

[5] F. Markatopoulou, G. Tsoumakas, and L. Vlahavas, *Instance-based ensemble pruning via multi-label classification*, in Proc. 22nd IEEE Int. Conf. Tools With Artif. Intell., vol. 1. Oct. 2010, pp. 401408.

[6] L. Kuncheva, *Switching between selection and fusion in combining classifiers: An experiment*, IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 32, no. 2, pp. 146156, Apr. 2002.

[7] A. Tsymbal, M. Pechenizkiy, and P. Cunningham, *Diversity in search strategies for ensemble feature selection*, Inf. Fusion, vol. 6, no. 1, pp. 8398, 2005.

[8]]M. Dash and H. Liu, *Feature selection for classification*, Intell. Data Anal., vol. 1, nos. 14, pp. 131156, 1997.

[9] R. Jensen and Q. Shen, *Computational Intelligence and Feature Selection: Rough and Fuzzy Approaches*, New York, NY, USA: Wiley-IEEE Press, 2008.

[10] M. Shah, M. Marchand, and J. Corbeil, *Feature selection with conjunctions of decision stumps and learning from microarray data*, IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 1, pp. 174186, Jan. 2012.

[11] E. P. Xing, M. I. Jordan, and R. M. Karp, *Feature selection for high dimensional genomic microarray data*, in Proc. 18th Int. Conf. Mach. Learn, 2001, pp. 601608.

[12] M. Dash and H. Liu, *Consistency-based search in feature selection*, Artif. Intell., vol. 151, nos. 12, pp. 155176, Dec. 2003.

[13] M. A. Hall, *Correlation-based feature selection for discrete and numeric class machine learning*, in Proc. 17th Int. Conf. Mach. Learn, 2000, pp. 359366.

[14] R. B. Bhatt and M. Gopal, *On fuzzy-rough sets approach to feature selection*, Pattern Recognit. Lett., vol. 26, no. 7, pp. 965975, 2005.

[15] R. Jensen and Q. Shen, *New approaches to fuzzy-rough feature selection*, IEEE Trans. Fuzzy Syst., vol. 17, no. 4, pp. 824838, Aug. 2009.

[16] N. Mac Parthalin, Q. Shen, and R. Jensen, *A distance measure approach to exploring the rough set boundary region for attribute reduction*, IEEE Trans. Knowl. Data Eng., vol. 22, no. 3, pp. 305317, Mar. 2010.

[17] R. W. Swiniarski and A. Skowron, *Rough set methods in feature selection and recognition*, Pattern Recognit. Lett., vol. 24, no. 6, pp. 833849, 2003.

[18] N. Mac Parthalin and R. Jensen, *Measures for unsupervised fuzzy rough feature selection*, Int. J. Hybrid Intell. Syst., vol. 7, no. 4, pp. 249259, Dec. 2010.

[19]]Z. Zhu, Y.-S. Ong, and M. Dash, *Wrapper-filter feature selection algorithm using a memetic framework*, IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 37, no. 1, pp. 7076, Feb. 2007.

[20] C.-N. Hsu, H.-J. Huang, and D. Schuschel, *The annigma-wrapper approach to fast feature selection for neural nets*, IEEE Trans. Syst., Man, Cybern. B, vol. 32, no. 2, pp. 207212, Apr. 2002.

[21] R. Kohavi and G. H. John, *Wrappers for feature subset selection*, Artif. Intell., vol. 97, no. 1, pp. 273324, 1997.

[22] H. Liu and H. Motoda, *Computational Methods of Feature Selection*, London, U.K.: Chapman Hall/CRC, 2007.

[23] J. Debuse and V. Rayward-Smith, *Feature subset selection within a simulated annealing data mining algorithm*, J. Intell. Inf. Syst., vol. 9, no. 1, pp. 5781, 1997.

[24] X. Wang, J. Yang, X. Teng, W. Xia, and R. Jensen, *Feature selection based on rough sets and particle swarm optimization*, Pattern Recognit. Lett., vol. 28, no. 4, pp. 459471, 2007.

[25] Z. W. Geem, *Recent Advances in Harmony Search Algorithm (Studies in Computational Intelligence)*, vol. 270. Berlin, Germany: Springer, 2010.

[26] K. S. Lee and Z. W. Geem, *A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice*, Comput. Meth. Appl. Mech. Eng., vol. 194, nos. 3638, pp. 39023933, Sep. 2005.

[27]]Diao, Ren, *Feature selection inspired classifier ensemble reduction*, IEEE transactions on cybernetics 44.8, (2014): 1259-1268.

[28] L. Nanni and A. Lumini, *Ensemblator: An ensemble of classifiers for reliable classification of biological data*, Pattern Recognit. Lett., vol. 28,no. 5, pp. 622630, 2007.

[29] M. A. Tahir, J. Kittler, and A. Bouridane, *Multilabel classification using heterogeneous ensemble of multi-label classifiers*, Pattern Recognit Lett., vol. 33, no. 5, pp. 513523, 2012.