Ensemble Pruning Using Harmony Search

Shina Sheen[1,*], S.V. Aishwarya[1], R. Anitha[1], S.V. Raghavan[2], and S.M. Bhaskar[3]

[1] PSG College of Technology, Coimbatore, India
shina_np12@yahoo.com, aishwarya.sv@gmail.com,
anitha_nadarajan@mail.psgtech.ac.in
[2] Indian Institute of Technology, Madras
svr@cs.iitm.ernet.in
[3] CCA&R, New Delhi
smb@nic.in

Abstract. In recent years, a number of works proposing the combination of multiple classifiers to produce a single classification have been reported. The resulting classifier, referred to as an ensemble classifier, is generally found to be more accurate than any of the individual classifiers making up the ensemble. In an ensemble of classifiers, it is hoped that each individual classifier will focus on different aspects of the data and error under different circumstances. By combining a set of so-called base classifiers, the deficiencies of each classifier may be compensated by the efficiency of the others. Ensemble pruning deals with the reduction of an ensemble of predictive models in order to improve its efficiency and performance. Ensemble pruning can be considered as an optimization problem. In our work we propose the use of Harmony search, a music inspired algorithm to prune and select the best combination of classifiers. The work is compared with AdaBoost and Bagging among other popular ensemble methods and our method is shown to perform better than the other methods. We have also compared our work with an ensemble pruning technique based on genetic algorithm and our model has shown better accuracy.

Keywords: Ensemble learning, Ensemble pruning, Harmony search, Classification.

1 Introduction

There is a large body of theoretical and empirical research showing that combining the predictions of a collection of classifiers can be an effective strategy to improve generalization performance [1-4]. The combination methods proposed in the literature are based on "voting" rules, statistical techniques, belief functions, and other "classifier fusion" schemes. As an example, the "majority" voting rule interprets each classification result as a "vote" for one of the data classes and assigns the input pattern to the class receiving the majority of votes. Such methods assume that, for each pattern, different classifiers make different classification errors. Ensembles

* Corresponding author.

E. Corchado et al. (Eds.): HAIS 2012, Part II, LNCS 7209, pp. 13–24, 2012.

composed of independent classifiers generally use equally weighted voting to produce a final class prediction. Given an unlabeled instance, the usual procedure is to query all classifiers in the ensemble and then output the majority class. In this work, we show that it is possible to estimate the outcome of the voting process with a specified confidence level by polling only a subset of the classifiers in the ensemble. We make use of Harmony search, a music inspired algorithm to identify the best subset of classifiers for a specific task. Using this procedure, only a subset of the predictors in the ensemble needs to be queried, which results in significant speed up of the classification process.

The ensemble method we propose in this paper comprises of three phases: the generation of multiple predictive models or classifiers, reduction of ensemble size prior to combination called as ensemble pruning and the combination of the final ensemble.

2 Ensemble of Classifiers

Classification is one task of data mining which allows predicting if a data instance is a member of a predefined class. Input is a training dataset S, where each instance is typically represented in the form of vector attributes $<x_1,x_2,x_3\ldots,x_n,y>$, y is the class attribute. The objective of classification is to train a classification algorithm A, on training data set S to find a good approximation of a certain function f(x)=y which is called the classifier. Evaluation of classifier accuracy is performed with a dataset T independent of S. The classifier will thereafter be able to predict the class y for new data d. An ensemble contains a number of classifiers called base learners. The generalization ability of an ensemble is usually much stronger than that of base learners. The various phases in an ensemble method is presented below.

2.1 Generating Models

Typically, an ensemble is constructed in two steps. First, a number of base learners are produced, which can be generated in a *parallel* style or in a *sequential* style where the generation of a base learner has influence on the generation of subsequent learners. Then, the base learners are combined to be used for prediction, where the most popular combination schemes are *majority voting* for classification and *weighted averaging* for regression. The basic principle in ensemble learning is to generate multiple versions of the classifier by perturbing the training set, construction method or some parameters. Several techniques have been used for this and the most notable among these are bagging[2], boosting[5] and random subspace method[7].

An ensemble can be composed of homogeneous or heterogeneous models. Our ensemble model comprises of heterogeneous models derived from running different learning algorithms on the same data set.

2.2 Pruning the Ensemble

Let $\Omega= \{M_1, \ldots..M_n\}$ be an ensemble of n classifiers. M_i is a classifier that can predict the class $M_i(x)$ of an observation x. The problem of ensemble pruning is to find the

best subset of Ω such that the combination of the selected classifiers will have the highest possible degree of accuracy.

The various ensemble pruning methods in literature generally fall into the following categories: Ranking based [8-11] and search based [12-15]. Prodromidis et al [8] suggested ranking classifiers according to their classification performance on a separate validation set and their ability to correctly classify specific classes. Similarly Caruana et al [9] presented a forward stepwise selection procedure in order to select the most relevant classifiers (that maximize the ensemble's performance) among thousands of classifiers. The algorithm FS-PP-EROS generates a selective ensemble of rough subspaces [10]. The algorithm performs an accuracy-guided forward search to select the most relevant members. Margineantu and Dietterich [11] presented an agreement based ensemble pruning which measures the Kappa statistics between any pair of classifiers. Pairs of classifiers are then selected in ascending order of their agreement level till the desired ensemble size is reached.

Rokach et al[12] suggested ranking the classifiers first according to their ROC performance and then evaluating the performance of the ensemble subset by using the top ranked members. Prodromidis and Stolfo [13] introduced a backwards correlation based pruning. The main idea is to remove the members that are least correlated to a meta-classifier which is trained based on the classifiers' outputs. In each iteration they remove one member and recompute the new reduced meta-classifier (with the remaining members). The meta-classifier in this case is used to evaluate the collective merit of the ensemble. Zhang et al. [14] formulated the ensemble pruning problem as a quadratic integer programming problem to look for a subset of classifiers that has the optimal accuracy-diversity trade-off. Using a semi-definite programming (SDP) technique, they efficiently approximated the optimal solution. The Gasen-b method [15] performed stochastic search in the space of model subsets using a genetic algorithm. The ensemble is represented as a bit string, using one bit for each model. Models are included or excluded from the ensemble depending on the value of the corresponding bit.

A detailed taxonomy of the various approaches is available in [16].

2.3 Combining the Models

Once the classifiers are built, various techniques can be used to combine the results of each classifier. The most cited in literature are the majority vote, the weighted vote and stacking[6]. In majority voting, each classifier outputs a class value and the class with most votes is the one proposed by the ensemble. In weighted voting, the models are not treated equally as each of them is associated with a coefficient (weight), usually proportional to its classification accuracy. Stacking is a method that combines models by learning a meta-level (or level-1) model that predicts the correct class based on the decisions of the base level (or level-0) models. This model is induced on a set of meta-level training data that are typically produced by applying a procedure similar to k-fold cross validation on the training data. The outputs of the base-learners for each instance along with the true class of that instance form a meta-instance. A meta-classifier is then trained on the meta-instances. When a new instance appears for classification, the output of the all base-learners is first calculated and then propagated to the meta-classifier, which outputs the final result.

3 Harmony Search

Harmony search is a music-based metaheuristic optimization algorithm[17-19]. It was inspired by the observation that the aim of music is to search for a perfect state of harmony. This harmony in music is analogous to finding the optimality in an optimization process. A musician always intends to produce a piece of music with perfect harmony. On the other hand, an optimal solution to an optimization problem should be the best solution available to the problem under the given objectives and limited by constraints. Both processes intend to produce the best or optimum.

The key concepts of Harmony Search(HS) algorithm are musicians, notes, harmonies and harmony memory. In most optimisation problems solvable by HS, the musicians are the decision variables of the function being optimised. The notes played by the musicians are the values each decision variable can take. The harmony contains the notes played by all musicians, or a solution vector containing the values for each decision attribute. The harmony memory contains harmonies played by the musicians, or a storage place for solution vectors. A more concrete representation of harmony memory is a two dimensional matrix, where the rows contain harmonies (solution vectors) and the number of rows are predefined and bounded by the harmony memory size. Each column is dedicated to one musician, and the entire column stores all the notes played by him in all harmonies, referred to as the note domain for each musician in this paper. Harmony Search (HS) mimicks the improvisation process of jazz musicians and tries to find the best harmony, i.e., the solution for a certain problem. Consider the problem of optimizing a function $f(x)$ subject to $x_i \in X_i; i = 1,2,\ldots..,n$ where X_i is the possible range for each variable with $x_i^L \leq x_i \leq x_i^U$ where x_i^L and x_i^U are the lower and upper bounds for each variable.

HS works as follows:

Step 1) **Defining the optimization problem and algorithm parameters:** In the first step, the optimization problem is specified as follows:

Minimize (or Maximize) $f(x)$
subjected to $x_i \in X_i$, $i = 1, 2, \ldots, n$.

Step 2) **HM initialization:** In this step, each component of each vector in the parental population (HM), which is of size HMS (Harmony memory size), is filled with random harmonies (solutions) generated according to the bounds of the decision variables x_i.

Step 3) **New harmony improvisation:** In this step, a new harmony vector $\vec{x} = (x_1, x_2, \ldots..., x_n)$ is generated based on three rules: i) memory consideration ii) pitch adjustment and iii) random selection. Generating a new harmony is called 'improvisation.' In the memory consideration, the value of the first decision variable x_1 for the new vector is chosen from any of the values already existing in the current

HM, i.e., from the set $(x_1^1, x_1^2, \ldots\ldots, x_1^{HMS})$, with a probability HMCR. The values of the other decision variables $x_2, \ldots\ldots, x_n$ are also chosen in the same manner. The HMCR, which varies between 0 and 1, is the rate of choosing one value from the previous values stored in the HM, while $(1 - HMCR)$ is the rate of randomly selecting a fresh value from the possible range of values. Every component obtained by the memory consideration is further examined to determine whether it should be pitch adjusted. This operation uses the parameter PAR (which is the rate of pitch adjustment) as follows:

$$x_i = \begin{cases} x_i \pm rand(0,1) \cdot \ bw \ \ with\ probability\ PAR \\ x_i\ with\ probability\ (1 - PAR) \end{cases}$$

where *bw* is an arbitrary distance bandwidth (a scalar number), and *rand*() is a uniformly distributed random number between 0 and 1. Evidently, Step 3 is responsible for generating new potential variation in the algorithm.

Step 4) **HM update:** If the new harmony vector $\vec{x} = (x_1, x_2, \ldots\ldots, x_n)$ is better than the worst harmony in the HM, judged in terms of the objective function value, the new harmony is included in the HM, and the existing worst harmony is excluded from the HM. This is actually the selection step of the algorithm where the objective function value is evaluated to determine if the new variation should be included in the population (HM).

Step 5) **Check stopping criterion:** If the stopping criterion (maximum NI iterations) is satisfied, the computation is terminated. Otherwise, steps 3) and 4) are repeated.

4 Harmony Search for Ensemble Pruning

The aim of this work is to develop a harmony search [18] based, stand alone, reusable search strategy that can find optimal combination of classifiers. We need to map each key concepts of HS into elements in ensemble pruning. There are obvious analogies such as: each classifier combination can be seen as a harmony, and the objective function can be the maximization of accuracy. In this approach we map musicians onto the available classifiers to be selected.

Table 1. Harmony

C1	C2	C3	C4	C5	C6
0	1	1	0	0	0

The note domain of each musician is then a binary value, indicating whether or not the corresponding classifier is included in the harmony and the actual harmony can be represented as a series of bits. For example, as shown in Table 1, a harmony {0,1,1,0,0,0} will translate into classifier subset {C2, C3}.

4.1 Defining the Optimization Problem

Given an ensemble $\Omega = \{M_1, \ldots..M_n\}$, a combination method C, and a training set S from a distribution D over the labeled instance space, the goal is to find an optimal subset $Z_{opt} \subseteq \Omega$ which minimizes the generalization error, over the distribution D of the classifiers in Z_{opt} constructed using method C.

Let $F_1, F_2, \ldots F_n$ be the fitness values measured by the general performance (accuracy) of the classifiers. The problem can be defined as

Maximize $\sum_{i=1}^{n} F_i x_i$
Subject to the constraint $x_i \in (0,1), i = 1,2, \ldots n$

4.2 Initialisation Step

The initialisation step involves filling the harmony memory with randomly generated harmonies, i.e. randomly generated bit sets as shown in table 1. The various parameters are HMS (harmony memory size), PAR(Pitch Adjusting rate) and NI (maximum number of iterations).The harmony memory size is a sensitive parameter. A large harmony memory will give each musician more notes to choose from when improvising a new harmony. However, it will require a longer initialization in order to fill up the harmony memory and hence, may lead to slower convergence.

Table 2. Parameter settings for the proposed model

Parameter	*Value*
HMS	6
PAR	0.1
HMCR	0.9
NI	20

The most significant use of HMCR is in terms of selecting a previously unselected classifier, or vice versa. Pitch adjustment is similar to the mutation operator in genetic algorithms. PAR is usually use 0.1 to 0.5 in most applications. The parameter settings used in our work is shown in Table 2.

4.3 New Harmony Improvisation and Memory Update

A new solution vector is created using the parameters HMCR and PAR. Based on HMCR one of the values in harmony memory is selected or an entirely new value from (0,1) is chosen. The chosen bit is flipped or not based on the value of PAR. If the i^{th} bit of this vector equals 1, then the i^{th} classifier is allowed to participate in classification; if the bit is a 0, then the corresponding classifier does not participate. Each resulting subset of classifiers is evaluated according to its classification accuracy on a set of testing data

using weighted majority voting. In order to improvise a new harmony, each musician randomly selects a value out of their note domain. Together, such selected values form the new bit set. This set is then translated back to a classifier subset and evaluated. If the evaluation score is higher than any of the classifier subsets in the harmony memory, it replaces the worst subset; otherwise, the new bit set is discarded. The process iterates until max iteration is reached.

4.4 Proposed Model

The proposed model is shown in the schematic diagram in Fig 1.The dataset D= $\{(x_i, y_i), i=1,2,,\ldots m)\}$, where x_i is a vector with feature values and y_i is the value of the target variable provided to the n classifiers C_1, C_2,….C_n. Harmony search is used to find out the most optimal set of classifiers for a given dataset.

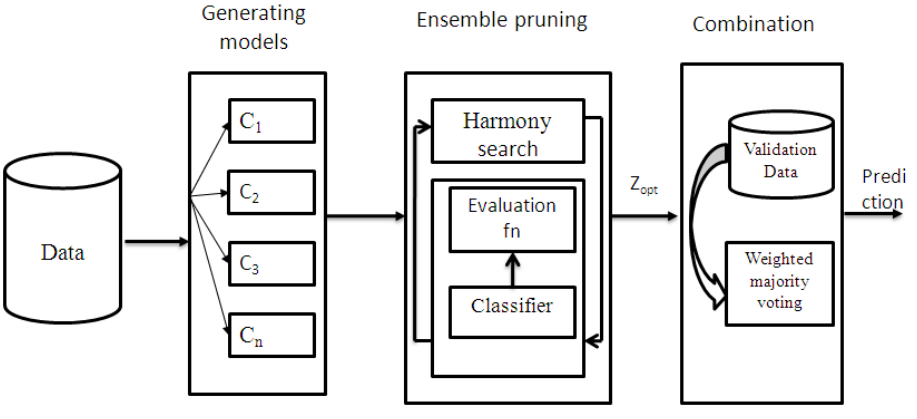

Fig. 1. Proposed model

The weak classifiers which don't contribute much to the final decision making process are eliminated at this stage. Ensemble combination is further done using weighted majority voting. The pseudo code for the proposed model HS_ENSEM is given in figure 2. The loop in line 2 trains the various classifiers once and the hypothesis is stored. Lines 14-17 constructs the harmony memory and generates binary solution vectors and calculates the fitness for each vector. The loop in line 20 creates a new solution vector based on the value of HMCR and PAR. The fitness of the new vector is calculated and it is compared with the values in harmony memory in line 28. If it is better than the worst harmony in memory it is replaced. After NI number of iterations the solution vector with the highest fitness is returned.

Algorithm HS_ENSEM
Input D: Dataset, Ω : Ensemble of Classifiers {C_1, ……., C_n}
Z_{opt} : Pruned ensemble set

1. Begin
2. For i = 1 to n
3. Train the classifiers with D and obtain hypothesis h
4. Endfor
5. Z_{opt} = harmonyensemble(Ω)
6. End
7. **Testing**
   a. Input an unlabeled instance X
   b. Evaluate X using Z_{opt}
   c. Obtain the composite hypothesis by weighted majority voting
   d. Choose the class with the maximum weighted votes.
8. **harmonyensemble(Ω)**
9. Begin
10. Initialize the parameters PAR, HMCR,NVAR,NI,BW,HMS
11. Int array X[];
12. Initialize X[]=0; t=0
13. Fitness=Accuracy of the ensemble
14. For i= 1 to HMS do
15. Generate random binary solutions and append it to HM
16. F[i] = Fitness value of the ith solution vector
17. End for
18. Worstfit = min(F[i])
19. While (t < NI)
20. For i=1 to NVAR
21. if (rand(0,1) < HMCR)
22. {
23. X[i] = HM[rand(0, HMS -1), i]
    a. if (rand(0,1) < PAR)
    b. flip the bit corresponding to X[i]
24. }
25. else
26. Randomly select the value of X[i] from population
27. end while
28. If fitness value of X[i] <= worstfit then
29. replace the vector in HM corresponding to worstfit with X[]
30. End if
31. End while
32. opt = max(F[i])
33. Z_{opt}= X[] corresponding to opt
34. Return Z_{opt}
35. End

Fig. 2. Pseudocode of the proposed model

The worst case time complexity of the algorithm is $O(n^2m)$ where 'n' is the number of classifiers and 'm' is the size of the data set. It has the same complexity as that of Genetic algorithm based pruning techniques but shows a better performance in terms of accuracy. The proposed method outperforms Hill climbing approach in terms of time which has a complexity of $O(n^2\ g(n;m))$, where $g(n;m)$ concerns the complexity of the evaluation process, which is linear with respect to 'm' and ranges from constant to quadratic with respect to 'n'.

5 Experimental Analysis

Six different algorithms implemented in Weka[20] were used to generate the independent heterogeneous classifiers for the experiments. The datasets considered are from the UCI machine learning repository [21]. The model was run for 20 iterations. Each harmony was evaluated with the accuracy as the objective function and the best harmonies were carried over to the next iteration. Larger harmony memory did not have an effect on performance.

The proposed method's accuracy was also compared to two commonly used ensemble techniques such as AdaBoost and Bagging and it is seen that it performs better in both instances.

In terms of computation performance and robustness, harmony search based approaches are computationally inexpensive themselves, because the algorithm comprises a very simple concept, and the implementation is also straightforward. The actual run time of the entire classifier ensemble process is then determined by two main factors, the max number of iterations, and the efficiency of the accuracy evaluation method.

We ran various experiments to test our approach. The following learning algorithms were used for the experiments: C4.5, PART, OneR, Naïve bayes, RBF Networks and Logistic Regression. Six data sets from the UCI repository were used. Each experiment was done using 10 fold cross validation for accuracy evaluation.

Table 3. Accuracy of the existing ensemble methods and our model

Ensemble methods	*Heart*	*Iris*	*Tictac*	*Kr-vs-kp*	*Pendigits*	*Satimage*
Bagging	81.25	94.00	92.068	99.123	99.2358	89.65
AdaBoost	**85.00**	95.33	72.547	93.836	20.4239	43.079
DECORATE	67.50	95.33	93.632	99.311	**99.7817**	89.113
Random forest	71.25	95.33	93.006	98.811	98.09	89.81
Random Subspace	58.28	98.657	88.309	99.780	99.09	88.958
Stacking	76.25	33.333	97.172	52.221	20.4076	23.9502
Our Model	76.88	**98.6577**	**97.178**	**99.780**	99.554	**92.140**

Tests were done using various state of the art ensemble combination techniques. The various techniques that we have compared our work with includes Bagging, AdaBoost, DECORATE, Random forest, Random Subspace and Stacking.The results are shown in table 3.

The accuracy of our proposed model is compared to that of EVEN [22] which is a genetic algorithm based ensemble pruning technique. Table 4 shows the results of comparing with their approach. We have used two data sets Pendigits and Satimage used in [22] and the results have been found to be encouraging.

The use of harmony memory in HS offers a major advantage over that of techniques like genetic algorithms, as it maintains a record of the historical data processed by previous iterations. All elements of the memory together contribute to the new harmony, while changes in genetic populations result in the destruction of previous knowledge of the problem. Harmony memory considering rate and pitch adjustment rate also help greatly in escaping from the local best solution.

Table 4. Comparison of our model with EVEN using Genetic algorithm

	Pendigits	*Satimage*
EVEN	99.21	89.94
Our Model	**99.554**	**92.17**

Fig 3 and Fig 4 shows the effect of HMCR and PAR on the accuracy of the model. It is seen that the best performance is for HMCR=0.9 and PAR=0.1.

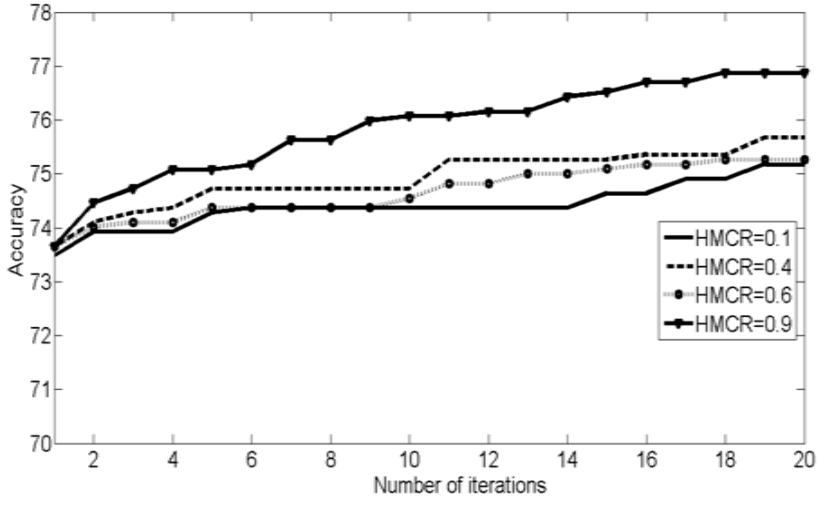

Fig. 3. Effect of HMCR on accuracy

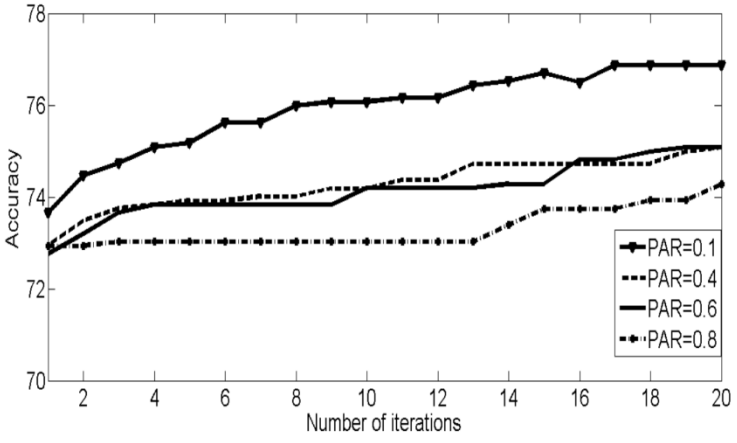

Fig. 4. Effect of PAR on accuracy

6 Conclusion and Future Work

A music inspired algorithm, harmony search is used to identify the most optimal ensemble of classifiers. The ensemble build using this model is found to be superior to the various state of the art techniques available today. It is also seen that ensemble learning provides a better performance than individual classifiers. Even though the computational cost of the proposed model is almost same as that of genetic algorithm based pruning, our model performs better in terms of accuracy.

As a possible extension of our work we plan to evaluate our method with the various evolutionary algorithms available and apply it for ensemble pruning in the domain of malware detection.

Acknowledgement. This work is a part of the Collaborative Directed basic Research on smart and secure environment project sponsored by NTRO, India.

References

1. Kuncheva, L.I.: Combining pattern classifiers: methods and algorithms. John Wiley & Sons Inc. (2004)
2. Breiman, L.: Bagging predictors. Mach. Learn. 24(2), 123–140 (1996)
3. Rodriguez, J.J., Kuncheva, L.I., Alonso, C.J.: Rotation forest: A new classifier ensemble method. IEEE Transactionson Pattern Analysis and Machine Intelligence 28(10), 1619–1630 (2006)
4. Banfield, R.E., Hall, L.O., Bowyer, K.W., Kegelmeyer, W.P.: A comparison of decision tree ensemble creation techniques. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(1), 173–180 (2007)

5. Freud, Y., Schapire, R.E.: Experiments with a New Boosting Algorithm. In: Saitta, L. (ed.) Machine Learning: Proceedings of the Thirteenth International Conference (ICML 1996), pp. 148–156. Morgan Kaufmann, San Francisco (1996)
6. Wolpert, D.H.: Stacked Generalization. Neural Networks 5(2), 241–259
7. Ho, T.K.: The Random Subspace Method for Constructing Decision Forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
8. Prodromidis, A.L., Stolfo, S.J., Chan, P.K.: Effective and efficient pruning of metaclassifiers in a distributed Data Mining system. Technical report CUCS-017-99, Columbia Univ. (1999)
9. Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble selection from libraries of models. In: Twenty-First International Conference on Machine Learning, Banff, Alberta, Canada, July 04-08 (2004)
10. Hu, Q., Yu, D., Xie, Z., Li, X.: EROS: Ensemble rough subspaces. Pattern Recognition 40, 3728–3739 (2007)
11. Margineantu, D., Dietterich, T.: Pruning adaptive boosting. In: Proc. Fourteenth Intl. Conf. Machine Learning, pp. 211–218 (1997)
12. Rokach, L., Arbel, R., Maimon, O.: Selective Voting - Getting More For Less in Sensor Fusion. International Journal of Pattern Recognition and Artificial Intelligence 20(3), 329–350 (2006)
13. Prodromidis, A.L., Stolfo, S.J.: Cost Complexity-Based Pruning of Ensemble Classifiers. Knowl. Inf. Syst. 3(4), 449–469 (2001)
14. Zhang, Y., Burer, S., Street, W.N.: Ensemble pruning via semi-definite programming. Journal of Machine Learning Research 7, 1315–1338 (2006)
15. Zhou, Z.H., Tang, W.: Selective Ensemble of Decision Trees. In: Wang, G., et al. (eds.) RSFDGrC 2003. LNCS (LNAI), vol. 2639, pp. 476–483. Springer, Heidelberg (2003)
16. Tsoumakas, G., Partalas, I., Vlahavas, I.: An Ensemble Pruning Primer. In: Okun, O., Valentini, G. (eds.) SUEMA 2009. SCI, vol. 245, pp. 1–13. Springer, Heidelberg (2009)
17. Geem, Z.W., Kim, J.H., Loganathan, G.V.: A new heuristic optimization algorithm: Harmony search. Simulation 76(2), 60–68 (2001)
18. Geem, Z.W. (ed.): Music-Inspired Harmony Search Algorithm: Theory and Applications. SCI. Springer, New York (2009)
19. Lee, K.S., Geem, Z.W.: A new metaheuristic algorithm for continuous engineering optimization: Harmony search theory and practice. Comput. Methods Appl. Mech. Eng. 194(36-38), 3902–3933 (2004)
20. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11(1) (2009)
21. http://repository.seasr.org/Datasets/UCI/arff/
22. Sylvester, J., Chawla, N.: Evolutionary Ensembles: Combining Learning Agents using Genetic Algorithms. In: Proc. of AAAI Workshop on Multi-agent Systems, pp. 46–51 (2005)