Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2014, Article ID 376950, 6 pages
http://dx.doi.org/10.1155/2014/376950

Research Article

A Weighted Voting Classifier Based on Differential Evolution

Yong Zhang, Hongrui Zhang, Jing Cai, and Binbin Yang

School of Computer and Information Technology, Liaoning Normal University, Dalian 116081, China

Correspondence should be addressed to Yong Zhang; zhyong@lnnu.edu.cn

Received 8 April 2014; Accepted 12 May 2014; Published 22 May 2014

Academic Editor: Caihong Li

Ensemble learning is to employ multiple individual classifiers and combine their predictions, which could achieve better performance than a single classifier. Considering that different base classifier gives different contribution to the final classification result, this paper assigns greater weights to the classifiers with better performance and proposes a weighted voting approach based on differential evolution. After optimizing the weights of the base classifiers by differential evolution, the proposed method combines the results of each classifier according to the weighted voting combination rule. Experimental results show that the proposed method not only improves the classification accuracy, but also has a strong generalization ability and universality.

1. Introduction

Ensemble learning is a new direction of machine learning, which trains a number of specific classifiers and selects some of them for ensemble. It has been shown that the combination of multiple classifiers could be more effective compared to any individual ones [1].

From a technical point of view, ensemble learning is mainly implemented as two steps: training weak base classifiers and selectively combining the member classifiers into a stronger classifier. Usually the members of an ensemble are constructed in two ways. One is to apply a single learning algorithm, and the other is to use different learning algorithms over a dataset [2]. Then, the base classifiers are combined to form a decision classifier. Generally, to get a good ensemble, the base learners should be as more accurate as possible and as more diverse as possible. So how to choose an ensemble of some accurate and diverse base learners is a focus of concern of many researchers [3].

In recent years, more and more researchers are concerned with ensemble learning [4]. There are many effective ensemble methods, such as boosting [5], bagging [6], and stacking [7]. Boosting is a method of producing highly accurate prediction rules by combining many "weak" rules which may be only moderately accurate. There are many boosting algorithms. The main variation between many boosting algorithms is their method of weighting training samples and hypotheses. AdaBoost is very popular and perhaps the most significant historically as it was the first algorithm that could adapt to the weak learners. Bagging trains a number of base learners each from a different bootstrap sample by calling a base learning algorithm. A bootstrap sample is obtained by subsampling the training dataset with replacement, where the size of a sample is the same as that of the training dataset. In a typical implementation of stacking, a number of first-level individual learners are generated from the training dataset by employing different learning algorithms. Those individual learners are then combined by a second-level learner which is called metalearner.

Among the most popular combination schemes, majority voting and weighted voting for classification are widely used. Simple majority voting is a decision rule that selects one of many alternatives, based on the predicted classes with the most votes. Majority voting does not require any parameter tuning once the individual classifiers have been trained [8, 9]. In case of weighted voting, weights of voting should vary among the different output classes in each classifier. The weight should be high for that particular output class for which the classifier performs well. So, it is a crucial issue to select the appropriate weights of votes for all the classes per classifier [2]. Weighting problem can be viewed as an optimization problem. Therefore, it can be solved by taking advantage of artificial intelligence techniques such as genetic algorithms (GA) and particle swarm optimization (PSO).

The existing literature shows the benefits of these methods of improving the classification performance [2].

Differential evolution (DE) is a simple, efficient, and population-based evolutionary algorithm for the global numerical optimization [10]. Due to its simple structure, ease of use, and robustness, DE has been successfully applied in many fields, including data mining, pattern recognition, digital filter design, and multiobjective optimization [11–13]. This paper describes a weighted voting ensemble learning scheme, in which the weight values of each base classifier are optimized by DE algorithm.

This paper is divided into five sections. Section 2 introduces the differential evolution. The proposed approach is presented in Section 3. Empirical studies, results, and discussions are presented in Section 4. Conclusions and future work are presented in Section 5.

2. Differential Evolution

Differential evolution algorithm was proposed by Storn and Price [10]. DE optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its simple formulae and then keeping whichever candidate solution that has the best score or fitness on the optimization problem at hand. DE algorithm starts with an initial population of N individuals: $X_{i,G}, i = 1, \ldots, N$, where the index i denotes the ith solution of the population at generation G. An individual is defined as a D-dimensional vector $X_{i,G} = [x_{i,G}(1), x_{i,G}(2), \ldots, x_{i,G}(j), \ldots, x_{i,G}(D)]$. There are three main operations of DE that are repeated till the stopping criterion is met. They are briefly described below.

Mutation. Mutation operation creates a donor vector $V_{i,G}$ corresponding to each population member or target vector $X_{i,G}$ in the current generation. The most frequently referred mutation strategies are presented below [14]:

DE/rand/1:

$$V_{i,G} = X_{r1,G} + F \times \left(X_{r2,G} - X_{r3,G}\right); \tag{1}$$

DE/best/1:

$$V_{i,G} = X_{\text{best},G} + F \times \left(X_{r1,G} - X_{r2,G}\right); \tag{2}$$

DE/current to best/1:

$$V_{i,G} = X_{i,G} + F \times \left(X_{\text{best},G} - X_{i,G}\right) + F \times \left(X_{r1,G} - X_{r2,G}\right); \tag{3}$$

DE/best/2:

$$V_{i,G} = X_{\text{best},G} + F \times \left(X_{r1,G} - X_{r2,G}\right) + F \times \left(X_{r3,G} - X_{r4,G}\right); \tag{4}$$

DE/rand/2:

$$V_{i,G} = X_{r1,G} + F \times \left(X_{r2,G} - X_{r3,G}\right) + F \times \left(X_{r4,G} - X_{r5,G}\right). \tag{5}$$

The indexes r_d, $d = 1, \ldots, 5$ represent the random and mutually different integers generated within the range $[1, N]$ and also different from index i. F is a mutation scaling factor within the range $[0, 2]$, usually less than 1. Vector $X_{\text{best},G}$ is the best individual vector with the best fitness in generation G.

Crossover. After mutation, crossover operation is performed between the target vector $X_{i,G}$ and its corresponding mutant vector $V_{i,G}$ to form the trial vector $T_{i,G} = [t_{i,G}(1), t_{i,G}(2), \ldots, t_{i,G}(j), \ldots, t_{i,G}(D)]$. For each j of the D variables,

$$t_{i,G}(j) = \begin{cases} v_{i,G}(j), & \text{if } \text{rand}_{i,j}[0,1] \le \text{CR or } j = j_{\text{rand}} \\ x_{i,G}(j), & \text{otherwise,} \end{cases} \tag{6}$$

where CR is a crossover control parameter, called crossover rate, within the range $[0, 1)$. $\text{rand}_{i,j}[0, 1]$ is a uniformly distributed random number, for each jth component of the ith vector. $j_{\text{rand}} \in [1, 2, \ldots, D]$ is a randomly chosen index, which ensures that $T_{i,G}$ gets at least one component from $V_{i,G}$.

Selection. After reproduction of the trial individual $T_{i,G}$, selection operation compares it to its corresponding target individual and decides whether the target or the trial individual survives to the next generation $(G + 1)$. The selection operation is described as

$$X_{i,G+1} = \begin{cases} T_{i,G}, & \text{if } f\left(T_{i,G}\right) \le f\left(X_{i,G}\right) \\ X_{i,G}, & \text{otherwise,} \end{cases} \tag{7}$$

where $f(\,)$ is the objective function to be optimized and ensures that a member of the next generation is the fittest individual. From (7), we can see that if the trial individual $T_{i,G}$ is better than target individual $X_{i,G}$, namely, $f(T_{i,G}) \le f(X_{i,G})$, then it replaces target individual in the next generation $(G + 1)$; otherwise it will continue with the target individual.

To improve optimization performance, DE algorithms are continually being developed. Many different strategies for performing crossover and mutation are proposed [15–18].

3. Our Proposed Approach

This section describes the proposed weighted voting ensemble learning method based on differential evolution (DEWVote). In our proposed method, we randomly select base classifiers. We find the proper weights of all the base classifiers depending on the prediction confidence through DE algorithm. The whole procedure is summarized in Figure 1.

3.1. Selection and Training of Base Classifiers. The use of ensemble of classifiers has gained wide acceptance in machine learning and statistics community due to significant improvement in accuracy. The individual classifiers should be as diverse as possible. In the well-known ensemble techniques such as bagging and boosting, such diversities are achieved by manipulating the training examples in order to generate multiple hypotheses. In our proposed approach, we select five

Input: The control parameters of DE: mutation factor F, crossover rate CR, and population size N.
(1) Initialization(); {Generate uniformly distributed random population of N individuals $X_G = \{X_{1,G}, X_{2,G}, \ldots, X_{i,G}, \ldots, X_{N,G}\}$, where $X_{i,G} = [x_{i,G}(1), x_{i,G}(2), \ldots, x_{i,G}(j), \ldots, x_{i,G}(D)]$ is a vector representing the weights $(w_1, w_2, \ldots, w_j, \ldots, w_D)$ of D base classifiers.}
(2) Set the generation iterator $G = 0$.
(3) **while** the stopping criterion is not satisfied **do**
(4) **for** $(i = 0; i < N; i++)$ **do**
(5) Select random indexesr_1, r_2, and r_3 to be different from each other and from the index i.
(6) Compute a mutant vector $V_{i,G}$ using (1).
(7) Generate random number j_{rand}.
(8) **for** $(j = 0; j < D; j++)$ **do**
(9) Decide trial individual $T_{i,G}$ using (6).
(10) **end for**
(11) Compute the fitness of the vector $T_{i,G}$ and $X_{i,G}$ using 10-fold cross validation, and update the vector $X_{i,G+1}$ of the next generation $(G + 1)$ using (7).
(12) **end for**
(13) Update generation iterator $G = G + 1$.
(14) **end while**
Output: The optimal weights $(w_1, w_2, \ldots, w_j, \ldots, w_D)$ for DEWVote.

Algorithm 1: DE algorithm for DEWVote's model selection.

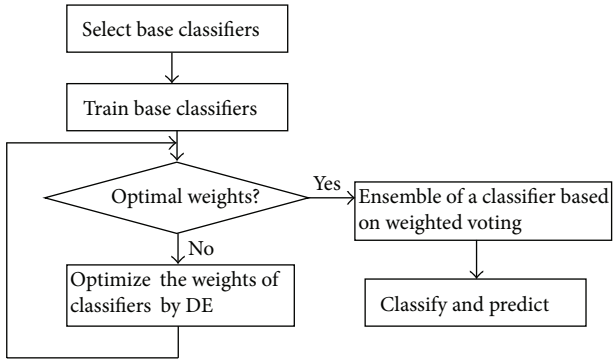

Figure 1: The whole procedure of our proposed approach.

base classifiers to learn, including C4.5, Naive Bayes, Bayes Nets, k-nearest neighbor (k-NN), and ZeroR.

3.2. DE-Based Model for Parameters Selection. In this section, we are concerned with the parameters selection for the proposed DEWVote. The parameters that should be optimized in DEWVote are the weights of each base classifier in an ensemble. Different parameters settings have a heavy impact on the performance of DEWVote. We select the differential evolution to search the optimal weights.

DE has a random initial population of solution candidates that is then improved using the evolution operations. In general, we employ the predefined maximum iterations $M_{\max}$ to determine the stopping criterion of DE. Other control parameters for DE are the mutation scaling factor $F \in (0, 1)$, the crossover rate CR $\in (0, 1)$, and the population size N. The process of the DE-based parameters selection for DEWVote is shown in Algorithm 1 with the following explanations.

Initialization. Initialize a population of N individuals: $X_G = \{X_{1,G}, X_{2,G}, \ldots, X_{i,G} \ldots, X_{N,G}\}$. An individual is defined as a D-dimensional vector: $X_{i,G} = [x_{i,G}(1), x_{i,G}(2), \ldots, x_{i,G}(j), \ldots, x_{i,G}(D)]$, which represents the weights $(w_1, w_2, \ldots, w_j, \ldots, w_D)$ of base classifiers and D is the size of base classifiers. Each individual is generated by the uniform distribution in the range $[0, 1]$.

Fitness Evaluation. Train the DEWVote by using each individual vector, and the corresponding 10-fold cross-validation accuracy is then evaluated as the fitness function.

Given the number of categories m and D base classifiers to vote, the prediction category c_k of weighted voting for each sample k is described as

$$c_k = \arg \max_j \sum_{i=1}^{D} \left(\Delta_{ji} \times w_i \right), \tag{8}$$

where Δ_{ji} is the binary variable. If the ith base classifier classifies sample k into the jth category, then $\Delta_{ji} = 1$; otherwise, $\Delta_{ji} = 0$. w_i is the weight of the ith base classifier in an ensemble, which is optimized by DE algorithm in Algorithm 1.

Then, the accuracy is defined as

$$\text{Acc} = \frac{\sum_k \{1 \mid c_k \text{ is the true category of sample } k\}}{\text{Size of test samples}} \times 100\%. \tag{9}$$

After obtaining the best individual by the differential evolution, namely, the optimal weight $(w_1, w_2, \ldots, w_j, \ldots, w_D)$,

Table 1: Summary of datasets.

Dataset	Total samples	Number of attributes	Number of classes
Breast-cancer	286	10	2
Contact-lenses	24	5	3
Credit-g	1000	21	2
Diabetes	768	9	2
Ionosphere	351	35	2
Iris	150	5	3
Iris.2D	150	3	3
Glass	214	10	7
Labor	57	17	2
Segment-challenge	1500	20	7
Soybean	668	36	19
Supermarket	4627	217	2
Unbalanced	856	33	2
Vote	435	17	2
Weather.numeric	14	5	2

we generate the ensemble classifier to classify the test datasets using (8).

4. Experimental Results and Analysis

In this section, we present and discuss, in detail, the results obtained by the experiments carried out in this research.

We run our experiments under the framework of Weka [19] using 15 datasets to test the performance of the proposed method. These datasets are from the UCI Machine Learning Repository [20]. Information about these datasets is summarized in Table 1. In DE algorithm, the choice of DE parameters can have a large impact on optimization performance. Selecting the DE parameters that yield good performance has therefore been the subject of much research. For simplicity, we set factor $F = 0.5$, crossover rate CR = 0.9, population $N = 20$, and maximum iteration number $M_{\max} = 100$.

We first compared the performance with four base classifiers, including C4.5, Naive Bayes, Bayes Nets, and k-nearest neighbor (k-NN).

C4.5 is an algorithm used to generate a decision tree developed by Quinlan [21]. C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy.

A Naive Bayes classifier [22] is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. Bayes theorem provides a way of calculating the posterior probability. Naive Bayes classifier assumes that the effect of the value of a predictor on a given class is independent of the values of other predictors.

Bayes Nets or Bayesian networks [23] are graphical representation for probabilistic relationships among a set of random variables. A Bayesian network is an annotated directed acyclic graph (DAG) that encodes a joint probability distribution. The nodes of the graph correspond to the random variables. The links of the graph correspond to the direct influence from one variable to the other.

k-NN is a type of instance-based learning or lazy learning. In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.

To obtain a better measure of predictive accuracy, we compare these methods using 10-fold cross-validation. The cross-validation accuracy is the average of the ten estimates. In each fold nine out of ten samples are selected to be training set, and the left one out of the ten samples is testing set. This process repeats 10 times so that all samples are selected in both training set and testing set. Table 2 shows the average accuracy values of four single methods.

From Table 2, we can see that each method outperforms other single methods in some datasets. Comparatively, C4.5 has more accuracies than other methods in 8 of all 15 datasets. It is noted that these base classifiers are more diverse.

To obtain a better measure of predictive accuracy, we also compare several ensemble methods using 10-fold cross-validation, such as bagging, AdaBoost, majority voting, and our DEWVote approach. In the DEWVote approach, we select five classifiers as base learners, including C4.5, Naive Bayes, Bayes Nets, k-nearest neighbor (k-NN), and ZeroR [19]. ZeroR is the simplest classification method which relies on the target and ignores all predictors. ZeroR classifier simply predicts the majority category (class). Although there is no predictability power in ZeroR, it is useful for determining a baseline performance as a benchmark for other classification methods. Majority voting selects the same base classifiers as our approach. A Naive Bayes classifier is employed as the base learning algorithm of bagging and

Table 2: Comparison of 4 single methods.

Dataset	C4.5	Naive Bayes	Bayes Nets	k-NN
Breast-cancer	**0.755**	0.717	0.720	0.724
Contact-lenses	**0.833**	0.708	0.708	0.792
Credit-g	0.705	0.754	**0.755**	0.720
Diabetes	0.738	0.743	**0.763**	0.702
Ionosphere	**0.915**	0.826	0.895	0.863
Iris	**0.960**	**0.960**	0.927	0.953
Iris.2D	**0.960**	**0.960**	0.947	**0.960**
Glass	0.668	0.486	**0.706**	**0.706**
Labor	0.737	**0.895**	0.877	0.825
Segment-challenge	0.957	0.811	0.904	**0.962**
Soybean	0.915	0.930	**0.933**	0.912
Supermarket	**0.637**	**0.637**	**0.637**	0.371
Unbalanced	**0.986**	0.908	**0.98**6	0.977
Vote	**0.963**	0.901	0.901	0.924
Weather.numeric	0.643	0.643	0.571	**0.786**

Table 3: Comparison of 4 ensemble methods.

Dataset	Bagging	AdaBoost	Majority voting	DEWVote
Breast-cancer	0.689	0.703	0.822	**0.879**
Contact-lenses	0.750	0.708	0.958	**0.980**
Credit-g	0.737	0.695	0.854	**0.973**
Diabetes	0.754	0.743	**0.852**	0.824
Ionosphere	0.926	0.909	0.969	**0.984**
Iris	0.947	0.953	0.987	**0.997**
Iris.2D	0.953	0.953	0.980	**0.993**
Glass	0.724	0.743	0.939	**0.962**
Labor	0.842	0.877	0.986	**0.988**
Segment-challenge	0.959	0.975	**0.981**	0.980
Soybean	0.862	0.928	0.963	**0.998**
Supermarket	0.637	0.749	0.637	**0.995**
Unbalanced	0.986	0.985	0.986	**0.998**
Vote	0.961	0.954	0.954	**0.995**
Weather.numeric	0.643	0.714	0.926	**0.973**

AdaBoost. Naive Bayes classifiers are generated multiple times by each ensemble method's own mechanism. The generated classifiers are then combined to form an ensemble.

We present the mean of 10-fold cross-validation accuracies for 15 datasets. The results of ensembles are shown in Table 3. DEWVote shows more accuracies than other ensemble methods besides in *Diabetes* and *Segment-challenge* datasets, while majority voting outperforms other ensemble methods in these two datasets. It is of note that majority voting has more accuracies than bagging and AdaBoost. Comparatively speaking, DEWVote and majority voting obtain better performance in majority datasets. However, bagging and boosting obtain better performance than majority voting in *vote* dataset.

5. Conclusions

In this paper we give a novel approach to optimal weights of base classifiers by differential evolution and present a weighted voting ensemble learning classifier. The proposed approach adopts ensemble learning strategy and selects several base learners, which are as more diverse as possible to each other, to combine an ensemble classifier. Each weight of base learner is obtained by differential evolution algorithm.

We have compared the performance with the three classical ensemble methods, as well as with four base classifiers. Experimental results have confirmed that our approach consistently outperforms the previous approaches. DEWVote searches the weights through iteration operations. So, it has

more cost than other ensemble methods. In our future work, we will concentrate on reducing the computational cost.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partly supported by National Natural Science Foundation of China (no. 61373127), the China Postdoctoral Science Foundation (no. 20110491530), and the University Scientific Research Project of Liaoning Education Department of China (no. 2011186).

References

[1] T. Dietterich, "Ensemble learning," in *The Handbook of Brain Theory and Neural Networks*, 2nd edition, 2002.

[2] A. Ekbal and S. Saha, "Weighted vote-based classifier ensemble for named entity recognition: a genetic algorithm-based approach," *ACM Transactions on Asian Language Information Processing*, vol. 10, no. 2, article 9, 37 pages, 2011.

[3] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[4] Z. H. Zhou, "Ensemble learning," in *Encyclopedia of Biometrics*, S. Z. Li, Ed., pp. 270–273, Springer, Berlin, Germany, 2009.

[5] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, part 2, pp. 119–139, 1997.

[6] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[7] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[8] L. I. Kuncheva and J. J. Rodríguez, "A weighted voting framework for classifiers ensembles," *Knowledge and Information Systems*, vol. 38, no. 2, pp. 259–275, 2014.

[9] L. Lam and C. Y. Suen, "Application of majority voting to pattern recognition: an analysis of its behavior and performance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 553–568, 1997.

[10] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[11] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[12] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer, Berlin, Germany, 2005.

[13] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Transactions on Systems, Man, and Cybernetics A: Systems and Humans*, vol. 38, no. 1, pp. 218–237, 2008.

[14] R. Mukherjee, G. R. Patra, R. Kundu, and S. Das, "Cluster-based differential evolution with Crowding Archive for niching in dynamic environments," *Information Sciences*, vol. 267, pp. 58–82, 2014.

[15] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[16] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[17] J. Brest and M. S. Maučec, "Self-adaptive differential evolution algorithm using population size reduction and three strategies," *Soft Computing*, vol. 15, no. 11, pp. 2157–2174, 2011.

[18] W. Gong, Z. Cai, and Y. Wang, "Repairing the crossover rate in adaptive differential evolution," *Applied Soft Computing*, vol. 15, pp. 149–168, 2014.

[19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[20] UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/.

[21] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, Calif, USA, 1993.

[22] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, no. 2-3, pp. 103–137, 1997.

[23] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, San Francisco, Calif, USA, 1988.