

Research Article

An Application of Classifier Combination Methods in Hand Gesture Recognition

Guan-Wei Wang,¹ Chunxia Zhang,² and Jian Zhuang¹

¹ School of Mechanical Engineering, Xi'an Jiaotong University, Shaanxi, Xi'an, 710049, China

² School of Mathematics and Statistics, Xi'an Jiaotong University, Shaanxi, Xi'an, 710049, China

Correspondence should be addressed to Chunxia Zhang, cxzhang@mail.xjtu.edu.cn

Received 5 May 2011; Revised 27 September 2011; Accepted 9 October 2011

Academic Editor: Gordon Huang

Copyright © 2012 Guan-Wei Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Hand gesture recognition is a topic in artificial intelligence and computer vision with the goal to automatically interpret human hand gestures via some algorithms. Notice that it is a difficult classification task for which only one simple classifier cannot achieve satisfactory performance; several classifier combination techniques are employed in this paper to handle this specific problem. Based on some related data at hand, AdaBoost and rotation forest are seen to behave significantly better than all the other considered algorithms, especially a classification tree. By investigating the bias-variance decompositions of error for all the compared algorithms, the success of AdaBoost and rotation forest can be attributed to the fact that each of them simultaneously reduces the bias and variance terms of a SingleTree's error to a large extent. Meanwhile, kappa-error diagrams are utilized to study the diversity-accuracy patterns of the constructed ensemble classifiers in a visual manner.

1. Introduction

Hand gesture language, being one type of sign languages, originates from deaf people communicating with each other. In order to convey a meaning, a person generally needs to simultaneously combine the shape, orientation, and movement of his hands. The complex spatial grammars of hand gesture language are markedly different from the grammars of spoken languages.

Apart from facilitating deaf people to express their thoughts more conveniently, hand gestures are also a very natural part of human communications for common people. In some special situations such as very noisy environments where speech is not possible, they can become the primary communication medium. With the rapid development of hand gesture language, it nowadays has been applied in many fields such as human-computer interaction, visual surveillance, and so on [1–3]. Thus, hand gesture recognition becomes a hot topic in artificial intelligence and computer vision with the goal to automatically interpret human hand

gestures via some algorithms. Nevertheless, the great variability in spatial and temporal features of a hand gesture, such as that in time, size, and position, as well as interpersonal differences, makes the recognition problem be very difficult. For instance, different subjects have different hand appearance and may sign gesture in different pace.

Recent works in hand gesture recognition tend to handle the spatial and temporal variations separately and therefore lead to two smaller areas, namely, static posture recognition [4–10] and dynamic action recognition [2, 3, 11, 12]. In static posture recognition, the pose or the configuration of hands should be recognized using texture or some other features. By contrast, hand action recognition tries to interpret the meaning of the movement using dynamic features such as the trajectory of hands and so on. In the current study, we will focus on hand posture classification and recognition.

As for hand posture recognition, Bedregal et al. [4] introduced a fuzzy rule-based method for recognizing the hand gestures of LIBRAS (the Brazilian Sign Language). The method utilizes the set of angles of finger joints for classifying hand configurations and classifications of segments of hand gestures for recognizing gestures based on the concept of monotonic gesture segment. Just et al. [5] applied an approach that has been successfully used for face recognition to the hand posture recognition. The used features are based on the modified census transform and are illumination invariant. To achieve the classification and recognition processes, a simple linear classifier is trained using a set of feature look-up tables. Kim and Cipolla [7] attempted to address gesture recognition under small sample size where direct use of traditional classifiers is inappropriate due to high dimensionality of input space. Through combining canonical correlation analysis with the discriminative functions and scale-invariant feature transform (SIFT), they developed a pairwise feature extraction method for robust gesture recognition. In the experiments using 900 videos of 9 hand gesture classes, the proposed procedure was seen to notably outperform support vector classifier and relevance vector classifier. Based on a hand gesture fitting procedure via a new self-growing and self-organized neural gas (SGONG) network, a new method for hand gesture recognition was proposed by Stergiopoulou and Papamarkos [8]. The main idea of this method is as follows. Initially, the region of the hand is detected by applying a color segmentation technique based on a skin color filtering procedure in the $YCbCr$ color space. Then, the SGONG network is applied on the hand area so as to approach its shape. Based on the output grid of neurons produced by the neural network, palm morphologic characteristics are extracted. These characteristics, in accordance with powerful finger features, allow the identification of the raised fingers. Finally, the hand gesture recognition is accomplished through a likelihood-based classification technique. The proposed system has been extensively tested with success. Furthermore, Flasiński and Myśliński [9] presented a novel method for recognizing hand postures of the Polish sign language based on a synthetic pattern recognition paradigm. The main objective is to construct algorithms for generating a structural graph description of hand postures that can be analyzed with the ETPL(k) graph grammar parsing model. The structural description generated with the designed algorithms is unique and unambiguous, which results in good discriminative properties of the method.

In recent years, classifier combination strategies are rapidly growing and enjoying a lot of attention from pattern recognition as well as many other various domains due to their potential to greatly increase the prediction accuracy of a learning system. So far, these techniques have been proven to be quite versatile in a broad field of real applications such as face recognition, sentiment classification, and so forth [13–15]. Compared with one single classifier, an ensemble classifier has advantages to handle a classification task which is difficult for traditional methods, to achieve much higher prediction accuracy, and so forth.

In the research works of hand gesture recognition, however, there is very little literature about the applications of ensemble classifier methods. Dinh et al. [6] proposed a hand gesture classification system which is able to efficiently recognize 24 basic signs of the American sign language. In the system, computational performance is achieved through the use of a boosted cascade of classifiers that are trained by AdaBoost and informative Harr wavelet features. To adapt to complex representation of hand gestures, a new type of feature was suggested. Some experimental results show that the proposed approach is promising. Burger et al. [10] suggested to apply a belief-based method for SVM (support vector machine) fusion to recognize hand shapes. Moreover, the method was integrated into a wider classification scheme which allows taking into account other sources of information, by expressing them in the belief theories' formalism. The experimental results have shown that the proposed method was better than the classical methods in avoiding more than 1/5 of the mistakes. In this paper, we will employ several classifier combination techniques to deal with this specific classification problem. On the basis of some hand gesture data at hand, the methods AdaBoost and rotation forest are seen to behave significantly better than all the other considered algorithms, especially a classification tree. The reasons for the success of AdaBoost and rotation forest are then investigated by analyzing the bias-variance decompositions of error for all the compared algorithms. Moreover, the diversity-accuracy patterns of each ensemble classifier are studied via kappa-error diagrams in a visual manner and some promising results are obtained.

The rest of the paper is organized as follows. In Section 2, some commonly used classifier combination methods are reviewed in brief. Section 3 describes our used hand gesture recognition data set. Some experiments are conducted in Section 4 to find out the best method to solve this specific classification task. In the meantime, the reasons for the better performance of some algorithms are also investigated. Finally, the conclusions of the paper are offered in Section 5.

2. Brief Review of Ensemble Methods

In this section, we will give a brief review of our later used classifier combination techniques. In an ensemble classifier, multiple classifiers which are generally referred to as base classifiers should be first generated by means of applying a base learning algorithm (also called base learner) to different distributions of the training data, and then the outputs from each ensemble member are combined with a classifier fusion rule to classify a new example. In order to construct an ensemble classifier having better prediction capability, its constituent members should be accurate which at the same time should disagree as much as possible. In other words, the diversity between base classifiers and their individual accuracy are two essential factors for building a good ensemble classifier. However, these two factors are contradictory in practice. Generally speaking, there is a tradeoff between diversity and accuracy: as the classifiers become more diverse, they must become less accurate; conversely, as they become more accurate, the diversity between them must turn to be lower. With respect to different ensemble classifier generation strategies, the difference between them mainly lies in how to achieve a better tradeoff between diversity and accuracy of their base classifiers.

In order to facilitate the following discussions, we first introduce some notations here. Denote by $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ a training set consisting of N observations, where $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,p})^T$ is a p -dimensional feature vector and y_i is a class label coming from the set $\Phi = \{\phi_1, \phi_2, \dots, \phi_J\}$. Meanwhile, let $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)^T$ be an $N \times p$ matrix containing

• **Input**
 A training set $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^N$; A base learner \mathcal{W} ; Number of iterations T ; A new data point x to be classified.

• **Training Phase**
 For $t = 1, \dots, T$
 (1) Utilize the corresponding technique (i.e., bootstrap sampling or randomly selecting features) to get a training set \mathcal{L}_t .
 (2) Provide \mathcal{L}_t as the input of \mathcal{W} (random forest has an additional randomness injection operation) to train base classifier C_t .
 EndFor

• **Output**
 – The class label for x predicted by the ensemble classifier C^* as

$$C^*(x) = \arg \max_{y \in \mathcal{D}} \sum_{t=1}^T I(C_t(x) = y),$$

where $I(\cdot)$ denotes the indicator function which takes value 1 or 0 depending on whether the condition of it is true or false.

Algorithm 1: The general algorithmic framework for bagging, random subspace, and random forest methods.

the training features and $\mathbf{Y} = (y_1, y_2, \dots, y_N)^T$ be an N -dimensional vector containing the class labels for the training data. Put in another way, the training set \mathcal{L} can be expressed as concatenating \mathbf{X} and \mathbf{Y} horizontally; that is, $\mathcal{L} = [\mathbf{X} \ \mathbf{Y}]$. Furthermore, let T be the number of base classifiers, and let C_1, C_2, \dots, C_T be the T classifiers to construct an ensemble classifier, say, C^* . Denote by \mathcal{W} the given base learning algorithm to train each base classifier.

Bagging [16], random subspace [17], and random forest [18] may be the three most intuitive and simplest ensemble learning methods to implement. These three methods share the same combination rule, that is, simply majority voting scheme, to combine the decisions of their base classifiers. They only differ in how to use the given training set to generate a diverse set of base classifiers. Breiman's *bagging*, an acronym of *bootstrap aggregating*, trains its base classifiers by applying a base learning algorithm to some bootstrap samples [19] of the given training set \mathcal{L} . Each bootstrap sample is generated by performing N extractions with replacement from \mathcal{L} . As a result, in each of the resulted training sets for constructing base classifiers, many of the original training examples may appear several times whereas others may never occur. The random subspace method [17] obtains different versions of the original training set by performing modifications in the feature space (i.e., randomly selecting some features) rather than in the example space like bagging. The different training sets are then provided as the input of the given learning algorithm to build its base classifiers. As for random forest [18], it is an ensemble technique by taking a decision tree algorithm [20] as its base learner. Beside utilizing bootstrap sampling to obtain different training sets like bagging, random forest tries to produce additional diversity between base classifiers by adding a randomization principle in the tree induction process, which randomly selects a feature subset of size K (a hyperparameter of random forest) at each nonterminal node and then chooses the best split among it. In Algorithm 1, we present the general algorithmic framework for these three ensemble methods.

Nowadays, boosting can be deemed as the largest algorithmic family in the domain of ensemble learning. Unlike bagging, whose base classifiers can be trained in parallel, boosting is a sequential algorithm in which each new classifier is built by taking into account the performance of the previously generated classifiers. AdaBoost [21, 22], due to its simplicity

• **Input**
A training set $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^N$; A base learner \mathcal{W} ; Number of iterations T ; A new data point x to be classified.

• **Training Phase**
Initialization: Set the weight distribution over \mathcal{L} as $D_1(i) = 1/N$ ($i = 1, 2, \dots, N$).
For $t = 1, \dots, T$

- (1) According to the distribution D_t , draw N training instances at random from \mathcal{L} with replacement to compose a new set $\mathcal{L}_t = \{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^N$.
- (2) Provide \mathcal{L}_t as the input of \mathcal{W} to train a classifier C_t , and then compute the weighted training error of C_t as
$$\varepsilon_t = \Pr_{i \sim D_t}(C_t(x_i) \neq y_i) = \sum_{i=1}^N I(C_t(x_i) \neq y_i) D_t(i), \quad (1)$$
where $I(\cdot)$ takes value 1 or 0 depending on whether the i th training instance is misclassified or by C_t or not.
- (3) If $\varepsilon_t > 0.5$ or $\varepsilon_t = 0$, then set $T = t - 1$ and abort loop.
- (4) Let $\alpha_t = (1/2) \ln((1 - \varepsilon_t)/\varepsilon_t)$.
- (5) Update the weight distribution D_t over \mathcal{L} as
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t}, & \text{if } C_t(x_i) = y_i \\ e^{\alpha_t}, & \text{if } C_t(x_i) \neq y_i, \end{cases} \quad (2)$$
where Z_t is a normalization factor being chosen so that D_{t+1} is a probability distribution over \mathcal{L} .

Endfor

• **Output**
– The class label for x predicted by the ensemble classifier C^* as
$$C^*(x) = \arg \max_{y \in \mathcal{D}} \sum_{t=1}^T \alpha_t I(C_t(x) = y).$$

Algorithm 2: The main steps for the AdaBoost algorithm.

and adaptability, has become the most prominent member in boosting family. AdaBoost works by constructing an ensemble of subsidiary classifiers by applying a base learner to successive derived training sets that are formed by either resampling from the original training set [21] or reweighting the original training set [22] according to a set of weights maintained over the training set. Initially, the weights assigned to each training example are set to be equal, and, in subsequent iterations, these weights are adjusted so that the weight of the instances misclassified by the previously trained classifiers is increased whereas that of the correctly ones is decreased. Thus, AdaBoost attempts to produce new classifiers that are able to better predict the “hard” examples for the previous ensemble members. After a sequence of classifiers is trained, which is then combined by weighted majority voting in the final decision. Algorithm 2 lists the main steps for the resampling version of AdaBoost which is utilized in our later experiments.

Based on principal component analysis (PCA), Rodríguez et al. [23] developed a novel ensemble classifier creation technique rotation forest and demonstrated that it outperforms several other ensemble methods on some benchmark classification data sets from the UCI repository [24]. With the aim to create training data for a base classifier, the feature set of \mathcal{L} is randomly split into many subsets, and PCA is applied to each subset. All principal components are retained in order to preserve the variability information in the data. Thus, some axis rotations take place to form new features for training a base classifier. The main idea of rotation forest is to simultaneously encourage diversity and individual accuracy within an ensemble classifier. Specifically, diversity is promoted by using PCA to do feature axis

- **Input**
A training set $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^N = [\mathbf{X} \ \mathbf{Y}]$; Number of input features M contained in each feature subset; A base learner \mathcal{W} ; Number of iterations T ; A new data point \mathbf{x} to be classified.
- **Training Phase**
For $t = 1, 2, \dots, T$
 - Calculate the rotation matrix \mathbf{R}_t^a for the t th classifier C_t
 - (1) Randomly split the feature set $F = \{X_1, X_2, \dots, X_p\}$ into K subsets $F_{t,k}$ ($k = 1, 2, \dots, K$).
 - (2) For $k = 1, 2, \dots, K$
 - (a) Select the columns of \mathbf{X} that correspond to the attributes in $F_{t,k}$ to compose a submatrix $\mathbf{X}_{t,k}$.
 - (b) Draw a bootstrap sample $\mathbf{X}'_{t,k}$ (with sample size smaller than that of $\mathbf{X}_{t,k}$, generally taken to be 75%) from $\mathbf{X}_{t,k}$.
 - (c) Apply PCA to $\mathbf{X}'_{t,k}$ to obtain a matrix $\mathbf{D}_{t,k}$ whose i th column consists of the coefficients of the i th principal component.
 - (3) EndFor
 - (4) Arrange the matrices $\mathbf{D}_{t,k}$ ($k = 1, 2, \dots, K$) into a block diagonal matrix \mathbf{R}_t .
 - (5) Construct the rotation matrix \mathbf{R}_t^a by rearranging the rows of \mathbf{R}_t so that they correspond to the original features in F .
 - Provide $[\mathbf{X}\mathbf{R}_t^a \ \mathbf{Y}]$ as the input of \mathcal{W} to build a classifier C_t .
- EndFor
- **Output**
– The class label for \mathbf{x} predicted by the ensemble classifier C^* as
$$C^*(\mathbf{x}) = \arg \max_{y \in \Phi} \sum_{t=1}^T I(C_t(\mathbf{x}\mathbf{R}_t^a) = y).$$

Algorithm 3: The detailed steps of rotation forest method.

rotation for each base classifier while accuracy is sought by keeping all principal components and also using the whole data set to train each base classifier. Here, we summarize the detailed steps of rotation forest in Algorithm 3. Note that in this algorithm, there is another parameter, namely, the number of features M contained in each feature subset, which should be specified in advance. For simplicity, suppose that M is a factor of p so that the features are distributed into K subsets with each of them containing M features. Otherwise, the K th feature subset will have $p - (K - 1)M$ features. According to the results reported by Rodríguez et al. [23], rotation forest with $M = 3$ performs satisfactorily, and it thus provides users a directive advice to choose suitable value for M .

Furthermore, Melville and Mooney [25] proposed a new meta-learner DECORATE (diverse ensemble creation by oppositional relabeling of artificial training examples) that can use any “strong” learner (one that provides high accuracy on the training data) to build a diverse ensemble. This algorithm is accomplished by adding different randomly constructed instances to the training set when building new ensemble members. The artificial constructed instances are given category labels that disagree with the prediction of the current ensemble, thereby directly increasing diversity when a new learner is trained on the augmented data and added to the ensemble. Based on the experimental results using the J48 (an open source Java implementation of the C4.5 algorithm in the Weka data mining tool) decision-tree induction as a base learner and the analysis of the cross-validated learning curves for DECORATE as well as some other ensemble methods, Melville and Mooney [25] found that DECORATE produces highly accurate ensembles that outperform bagging, AdaBoost, and random forest low on the learning curve. In order to make this paper self-sufficient, we list the pseudocodes of DECORATE in Algorithm 4 as follows.

- **Input:**

\mathcal{L} : training set consisting of N instances;

\mathcal{W} : base learner whose output is assumed to be a class probability distribution;

C_{size} : desired ensemble size;

I_{max} : maximum number of iterations to construct an ensemble classifier;

R_{size} : a factor to determine number of artificial instances to generate.

- **Training phase**

- Initialization:

- Let $i = 1$ and trials = 1;

- Provide the given training set \mathcal{L} as the input of base learner \mathcal{W} to get a classifier C_i ;

- Initialize ensemble set $C^* = \{C_i\}$;

- Compute ensemble error as

$$\varepsilon = \frac{1}{N} \sum_{i=1}^N I(C^*(x_i) \neq y_i). \quad (3)$$

- While $i < C_{\text{size}}$ and trials $< I_{\text{max}}$

- (1) Generate $[R_{\text{size}} \times N]$ training instances, \mathcal{R} , according to the distribution of training data;

- (2) Label each instance in \mathcal{R} with probabilities that each class label is selected being inversely proportional to those predicted by C^* ;

- (3) Combine \mathcal{L} with \mathcal{R} to get a new training set \mathcal{L}' ;

- (4) Apply base learner \mathcal{W} to \mathcal{L}' to obtain a new classifier C' ;

- (5) Add C' to ensemble set C^* , namely, let $C^* = C^* \cup \{C'\}$;

- (6) Based on the training set \mathcal{L} , compute the ensemble error of C^* , say, ε' , as that done in equation (3);

- (7) If $\varepsilon' \leq \varepsilon$, let $i = i + 1$ and update ensemble error as $\varepsilon = \varepsilon'$; Otherwise, delete C' from the ensemble set C^* , that is, $C^* = C^* - \{C'\}$;

- (8) trials = trials + 1;

- EndWhile

- **Prediction phase**

- Let $p_{i,j}(x)$ be the probability that x comes from class j supported by the classifier C_i .

- Calculate the confidence for each class by the mean combination rule, that is,

$$d_j(x) = \frac{1}{L} \sum_{i=1}^L p_{i,j}(x), \quad j = 1, 2, \dots, J, \quad (4)$$

- where L stands for the real ensemble size.

- Assign x to the class with the largest confidence.

Algorithm 4: The pseudocodes for DECORATE ensemble method.

As can be seen in Algorithm 4, DECORATE builds an ensemble classifier iteratively like all the other ensemble methods. Initially, a classifier is trained on the basis of the given training data \mathcal{L} . In each successive iteration, one classifier is created by applying a base learner \mathcal{W} to \mathcal{L} combined with some artificial data. In each iteration, some artificial training instances are generated according to the data distribution that the given training data (only consider the input variables now) come from, where the number of instances to be generated is specified as a fraction, R_{size} , of the training set size N . As for the labels for each artificially generated training instance x_k , first utilize the current ensemble to predict the class membership probabilities $\hat{\mathbf{P}}(x_k) = (\hat{P}_1(x_k), \hat{P}_2(x_k), \dots, \hat{P}_J(x_k))^T$ that this instance belongs to each class. Then, replace zero probabilities with a small nonzero value and normalize the probabilities to make them form a probability distribution $\tilde{\mathbf{P}}(x_k) = (\tilde{P}_1(x_k), \tilde{P}_2(x_k), \dots, \tilde{P}_J(x_k))^T$. The label y_k of the instance x_k can then be determined such that the probability for each class i being selected is inversely proportional to the ensemble's prediction; namely, $\tilde{P}'_i(x_k) = (1/\tilde{P}_i(x_k)) / \sum_{j=1}^J (1/\tilde{P}_j(x_k))$. The main purpose in doing so is

to make the labels for the artificially generated instances differ maximally from the current ensemble's predictions in order to promote the diversity in the constructed ensemble classifier. Thus, the labeled artificially created training set is called *diversity data*. On the other hand, DECORATE tries to maintain the accuracy of each ensemble member while forcing diversity through rejecting a new classifier if adding it to the existing ensemble decreases its accuracy. The above whole process is repeated until the desired ensemble size is reached or the maximum number of iterations is exceeded.

It is worth to mention that, in DECORATE, the artificially generated training data are randomly picked from an approximation of the training-data distribution. For a numeric feature, the values of it are created from a Gaussian distribution whose mean and standard deviation are computed from the corresponding data in the training set. As for a nominal feature, the probability of occurrence of each distinct value in its domain should be first calculated in which the Laplace smoothing needs to be employed so that nominal feature values not represented in the training set still have a nonzero probability of occurrence. Then, some values can be generated based on this distribution. Another issue that should be pointed out is that we can only specify a desired ensemble size C_{size} when using DECORATE to deal with a classification task. The size L of the finally obtained ensemble may be smaller than C_{size} because the algorithm will terminate if the number of iterations exceeds the maximum limit even if C_{size} is not reached. As for R_{size} , it can take any value in theory. Nevertheless, the experiments done by Melville and Mooney [25] have shown that R_{size} lower than 0.5 adversely affect the performance of DECORATE, and the results with R_{size} chosen in range 0.5~1 do not vary much.

3. Data Set

For a hand gesture recognition problem, the task is to design a classifier to recognize different hand gestures where each gesture has a meaning of one simple or compound word. Our used data set contains 120 different signs of the Dutch sign language, each performed by 75 different persons. The images were captured at 640×480 pixels and 25 frames per second. Most sign examples include partial occlusions of hands of each other or with the face/neck. For the detailed process to obtain the experimental data, the readers can refer to [3, 26]. The supplementary video can be found on the Computer Society Digital Library at <http://www.computer.org/portal/web/csdl/doi/10.1109/TPAMI.2008.123>. We briefly introduce the process to collect the experimental data as follows. When a person is making a hand gesture, two cameras are used to independently record the continuous activity of his left and right hands. Because the gesture is made continuously and we would like to obtain some features, three images (frames) which, respectively, denote the beginning, middle, and ending of the gesture were extracted from the video recorded by one camera, and they were denoted as image 1, 2, and 3 here. Based on each obtained image, some segmentation algorithm was first used to segment two hands from the background, and then 7 invariant moments were computed for each hand. This process was repeated for the video of the other camera. Finally, we obtained 84 features in total through collecting the computed moments corresponding to two hands, three images, and two cameras together. In the original data set, there are totally 120 different gestures among which 29 ones denote compound words and 91 ones indicate simple words. For each gesture, there are 75 objects made by different persons, and every object is described by 84 features which were extracted in the above-mentioned way.

Unfortunately, we encountered a problem during segmenting two hands from one image. Sometimes there may occur an overlap between head, left hand, or right hand, which make it impossible to compute meaningful moments for each hand. Therefore, some corresponding features cannot be obtained in the general manner, and they are indicated as missing. In our experiments, we only considered the objects without missing features. Meanwhile, we tried to select the classes consisting of approximately equal number of objects. Through preprocessing the experimental data in this way, we finally obtained a set having 11 classes with about 70 objects in each class. The data set contains 793 objects in total, and each object is described by 84 features.

4. Experimental Study

4.1. Experimental Setting

In this section, we did some experiments by applying six commonly used classifier combination methods to the hand gesture recognition data set that is described previously. The considered ensemble methods include bagging [16], random forest [18], random subspace [17], AdaBoost [22], rotation forest [23], and DECORATE [25].

The experimental settings were as follows. In all the ensemble methods, a decision tree [20] was always adopted as the base learning algorithm because it is sensitive to the changes in its training data and can still be very accurate. The following experiments were all conducted in Matlab software with version 7.7. The decision tree algorithm was realized by the “Treefit” algorithm contained in the “Stats” package of Matlab. The parameters involved in this algorithm, such as the minimum number of training instances that impure nodes to be split should have, were all set to the default values. The implementations of the considered ensemble methods were realized in Matlab by writing programs according to their respective pseudocodes.

The ensemble size was set to 25 since the largest error reduction achieved by ensemble methods generally occurs at the first several iterations. Although larger ensemble size may result in better performance, the improvement achieved at the cost of additional computational complexity is trivial in comparison with that obtained with just a few iterations. As for the hyperparameter K in random forest, which specifies how many features should be firstly selected at each nonterminal node in the process of building a decision tree, the value of it was taken to be $\lceil \log_2(p) + 1 \rceil$ since some experiments [18] have proven that this choice makes random forest achieve good performance very often. When using random subspace technique to construct an ensemble classifier, one half of features were randomly selected to train its each constituent member. With respect to the parameter M which indicates the number of features contained in each feature subset in rotation forest, we set it to be 3 just like Rodríguez et al. [23] did because they have found that this value was almost always the best choice in their experiments. In DECORATE algorithm, the used parameters except for the ensemble size were all identical to those utilized by Melville and Mooney [25]; namely, the maximum number of iterations I_{\max} to build an ensemble classifier was set to 50, and the factor R_{size} to determine number of artificial examples to generate was chosen to be 1. Here, it should be noted that we can only specify a desired ensemble size for DECORATE algorithm while it may terminate if the number of iterations exceeds the maximum limit even if the desired ensemble size is not reached.

Table 1: The means and standard deviations (std.) of prediction errors, as well as scores for each classification method on the considered data set.

Statistic	SingleTree	Bagging	RandForest	RandSubspace	AdaBoost	DECORATE	RotForest
Mean error	45.85	32.09	32.80	32.31	29.21	40.39	27.55
Std.	1.19	1.03	1.03	0.95	1.01	2.73	1.10
Score	-6	0	0	0	4	-4	6

4.2. Results and Discussion

4.2.1. Comparison of Prediction Error

Because there have not been separate training and testing data to use, we employed the 10-fold cross-validation method to investigate the performance of the considered classification methods. Specifically, the data was first split into ten subsets with approximately equal sizes, and then nine of them were utilized as a training set to construct a forest while the other one was used to estimate the prediction error of the forest. The experiment was conducted ten times through alternating the role of ten subsets until each of them was used for testing once. We repeated the above process ten times with different random number generating seeds to split the data in order to eliminate the impact of random factor to the performance of each algorithm.

Before utilizing each obtained training set to carry out experiments, we preprocessed the data based on one normalization technique. Given a training set $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^N$, the normalization of the values corresponding to each feature $X_j (j = 1, 2, \dots, p)$ can be expressed as

$$\begin{aligned} \bar{x}_j^c &= \sum_{i=1}^{N_c} x_{ij}, & s_j^c &= \frac{1}{N_c - 1} \sum_{i=1}^{N_c} (x_{ij} - \bar{x}_j^c)^2, \\ s &= \sum_{c=1}^m P(\omega_c) s_j^c, & x'_{ij} &= \frac{x_{ij} - \bar{x}_j^c}{\sqrt{s}}, \quad i = 1, 2, \dots, N_c, \end{aligned} \quad (4.1)$$

where \bar{x}_j^c and s_j^c , respectively, denote the mean and variance for class ω_c and s is the weighted sum of the variances for each class with weights equal to class prior probabilities. After obtaining \bar{x}_j^c and s , the same mapping was applied to the test set.

Table 1 reported the mean as well as the standard deviation of the computed test errors for each algorithm. In order to make a complete comparison, the results calculated with a classification tree were also taken into account. In Table 1, the best results were highlighted in bold face to facilitate the comparison. With the aim to make clear whether there is significant difference between the performance of our evaluated ensemble methods on this specific data set, we adopted a one-tailed paired t -test with significance level $\alpha = 0.01$ to carry out some statistical tests between each pair of algorithms. If an algorithm is found to be significantly better than its competitor, we assigned score 1 to the former and -1 to the latter. If there is no significant difference between the two compared methods, they both score 0. Obviously, the higher the score of an approach, the better its performance. In the third row of Table 1, we listed the scores that each classification method gets according to the number of times that it has been significantly better or worse than the other algorithms.

From the obtained mean test errors and scores for each classification method, it can be observed that the prediction error of a single decision tree has been improved greatly by

each classifier combination technique, especially by AdaBoost and rotation forest. Among the ensemble algorithms, DECORATE was seen to perform much worse than the other ensemble learning strategies; the reason may be that its main advantage is to deal with classification problems with small training set size while the sample size of the current hand gesture recognition data set is medium. Based on the scores calculated from the statistical tests between each pair of algorithms, rotation forest is seen to be the best method to solve this specific problem, and it performs significantly better than all the other algorithms at significance level $\alpha = 0.01$. Meanwhile, AdaBoost is the second best approach since it was only beaten by rotation forest. However, a single decision tree and DECORATE behave very badly and they should not be selected to deal with this problem. As for the other three ensemble methods, the performance of them is almost equivalent even though their working mechanism is different as described in Section 2.

4.2.2. Bias-Variance Decomposition of Error

In order to investigate the reasons for the better performance of an ensemble classifier than its constituent members, to decompose its error into bias and variance terms is a good choice, and this method has been used by many researchers [27–29]. The decomposition of a learning machine's error into bias and variance terms originates from analyzing learning models with numeric outputs in regression problems. Given a fixed target and training set size, the conventional formulation of the decomposition breaks the expected error into the sum of three nonnegative quantities.

- (i) *Intrinsic "target noise"* (σ^2). This quantity is a lower bound on the expected error of any learning algorithm. It is the expected error of the Bayes optimal classifier.
- (ii) *Squared "bias"* (bias^2). This quantity measures how closely the learning algorithm's average guess (over all possible training sets of the given size) matches the target.
- (iii) *"Variance"* (variance). This quantity measures how much the learning algorithm's guess fluctuates from the target for the different training sets of the given size.

Notice that the above decomposition cannot be directly translated to contexts where the value to be predicted is categorical; a number of ways to decompose error into bias and variance terms in the field of classification prediction tasks have been proposed [30–33]. Each of these definitions is able to provide some valuable insight into different aspects of a learning machine's performance. In order to gain more insight into the performance of the considered ensemble methods on the hand gesture recognition data set, we utilized the bias-variance definition developed by Kohavi and Wolpert [30] in the current research, and they were, respectively, denoted by Bias and Var in the following discussions.

If denote by Y_H and Y_F the random variables, respectively, representing the evaluated and true labels of an instance, Bias and Var defined for a testing instance (\mathbf{x}, y) can be expressed as

$$\begin{aligned} \text{Bias}(\mathbf{x}) &= \frac{1}{2} \sum_{y' \in \Phi} \left[\Pr(Y_F = y' | \mathbf{x}) - \Pr(Y_H^{\ell} = y' | \mathbf{x}) \right]^2, \\ \text{Var}(\mathbf{x}) &= \frac{1}{2} \left\{ 1 - \sum_{y' \in \Phi} \left[\Pr(Y_H^{\ell} = y' | \mathbf{x}) \right]^2 \right\}. \end{aligned} \tag{4.2}$$

Table 2: The bias-variance decompositions for each classification method on the considered data set.

Statistic	SingleTree	Bagging	RandForest	RandSubspace	AdaBoost	DECORATE	RotForest
Mean error	45.85	32.09	32.80	32.31	29.21	40.39	27.55
Bias	26.01	22.17	21.38	22.47	18.05	22.94	19.23
Var	19.84	9.93	11.41	9.84	11.15	17.45	8.32

Here, the superscript \mathcal{L} is used in $Y_H^{\mathcal{L}}$ to denote that the evaluated class label is predicted by the machine trained on the set \mathcal{L} . The term $\Pr(\cdot)$ in the above formulae can be computed as the frequency that the event included in the parentheses occurs in the trials which are conducted with different training sets of the given size.

To compute the above two statistics, the distribution of where the training data come from should be known in advance. Unfortunately, the knowledge we have in the current hand gesture recognition situation is only a learning sample with medium size. In consequence, the Bias and Var terms should be estimated instead. In our experiments, the method similar to that used by [32], that is, ten trials of 10-fold cross-validation procedure, was utilized to estimate the bias and variance defined above. Once the cross-validation trials have been completed, the relevant measures can be estimated directly from the observed distribution of results. The use of cross-validation in this way has the advantage that every instance in the available data \mathcal{L} is used the same number of times, both for training and for testing.

According to the above approach, the previously defined bias and variance decompositions of the errors for each classification method were estimated for each instance in the given data set, and then their values were averaged over that data set. Detailed decompositions of mean error into Bias and Var for each classification method were provided in the following Table 2.

As can be seen from Table 2, the order of the considered classification methods ranked in terms of Bias value from best to worst is AdaBoost, RotForest, RandForest, Bagging, RandSubspace, DECORATE, and SingleTree. With regard to Var, these algorithms are rated from best to worst as RotForest, RandSubspace, Bagging, AdaBoost, RandForest, DECORATE, and SingleTree. Therefore, the better performance of RotForest and AdaBoost can be attributed to the fact that they reduce both bias and variance of the SingleTree's error to a large degree. RotForest does a better job to reduce variance term while AdaBoost has a small advantage on reducing bias. The working mechanism of RandForest is similar to that of RotForest and AdaBoost, but the reduction achieved by it is not enough. In the meantime, bagging and RandSubspace are observed to mainly reduce the variance term. As for DECORATE, it was seen to only decrease the bias and variance of the SingleTree's error to a small extent.

4.2.3. Kappa-Error Diagrams

On the other hand, many researchers [34–36] have pointed out that the success of an ensemble classifier achieving much lower generalization error than its any constituent member lies in the fact that the ensemble classifier consists of highly accurate classifiers which at the same time disagree as much as possible. Put in another way, with the purpose to construct an ensemble classifier with good performance, we should achieve a good tradeoff between diversity and accuracy.

The kappa-error diagrams developed by Margineantu and Dietterich [37] provide us an effective means to visualize how an ensemble classifier which is constructed by some ensemble learning technique attempts to reach the tradeoff between the diversities and accuracies of its constituent members. For each pair of classifiers, the diversity between them is measured by the statistic kappa (κ) which evaluates the level of agreement between two classifier outputs while correcting for chance; the accuracy of them is measured by the average of their error rates estimated on the testing data set. A kappa-error diagram is a scatter plot in which each point corresponds to a pair of classifiers C_i and C_j . On the x -axis of the plot is the diversity value κ and on the y -axis of it is the mean error of C_i and C_j , say, $E_{i,j} = (E_i + E_j)/2$.

The statistic κ is defined as follows. Suppose that there are J classes and κ is defined on the $J \times J$ coincidence matrix \mathcal{M} of two classifiers C_i and C_j ($i, j = 1, 2, \dots, T$) where T is the number of classifiers in an ensemble. The entry $m_{k,s}$ of \mathcal{M} is the proportion of the testing data set, in which classifier C_i labels as ω_k and classifier C_j labels as ω_s . Then the agreement between C_i and C_j can be measured as

$$\kappa_{i,j} = \frac{\sum_{k=1}^J m_{kk} - ABC}{1 - ABC}, \quad (4.3)$$

where $\sum_{k=1}^J m_{kk}$ is the observed agreement between the two classifiers C_i and C_j . "ABC," the acronym of "agreement-by-chance," is defined as

$$ABC = \sum_{k=1}^J \left(\sum_{s=1}^J m_{k,s} \right) \left(\sum_{s=1}^J m_{s,k} \right). \quad (4.4)$$

According to the above definition of κ , low values of it indicate higher diversity. And since small values of $E_{i,j}$ indicate better accuracy, the most desirable pairs of classifiers should lie in the bottom left corner of the scatter plot.

Figure 1 illustrates the kappa-error diagrams of the ensemble classifiers constructed by each ensemble algorithm on the hand gesture recognition data. All the constructed forests but the one built by DECORATE consist of 25 trees; therefore there are 300 (C_{25}^2) points in each plot. Because the data set has no separate training and testing parts, we randomly took 90% of observations to build the forests and the remaining to compute the kappa-error diagrams. With regard to each plot, the axes of them were adjusted to be identical so that the comparisons can be easily carried out. Moreover, on the top of each plot in Figure 1, we presented the used ensemble method, its prediction error estimated on the testing set, and the coordinates (shown as the red point in each plot) for the mean of diversities and that of errors which were averaged over all pairs of base classifiers.

As can be observed in Figure 1, DECORATE gives a very compact cloud of points. Each point has a low error rate and a high value for κ , which indicates that the base classifiers are accurate but not very diverse. The shape of the kappa-error diagrams for bagging and random subspace is similar, but the points for bagging are more diverse while those for random subspace are slightly more accurate. Although the test error of random forest is identical to that of AdaBoost, we can find that AdaBoost is more diverse but is not as accurate as random forest. In the meantime, rotation forest is seen to achieve better diversity than random subspace and DECORATE, but its accuracy is only worse than DECORATE.

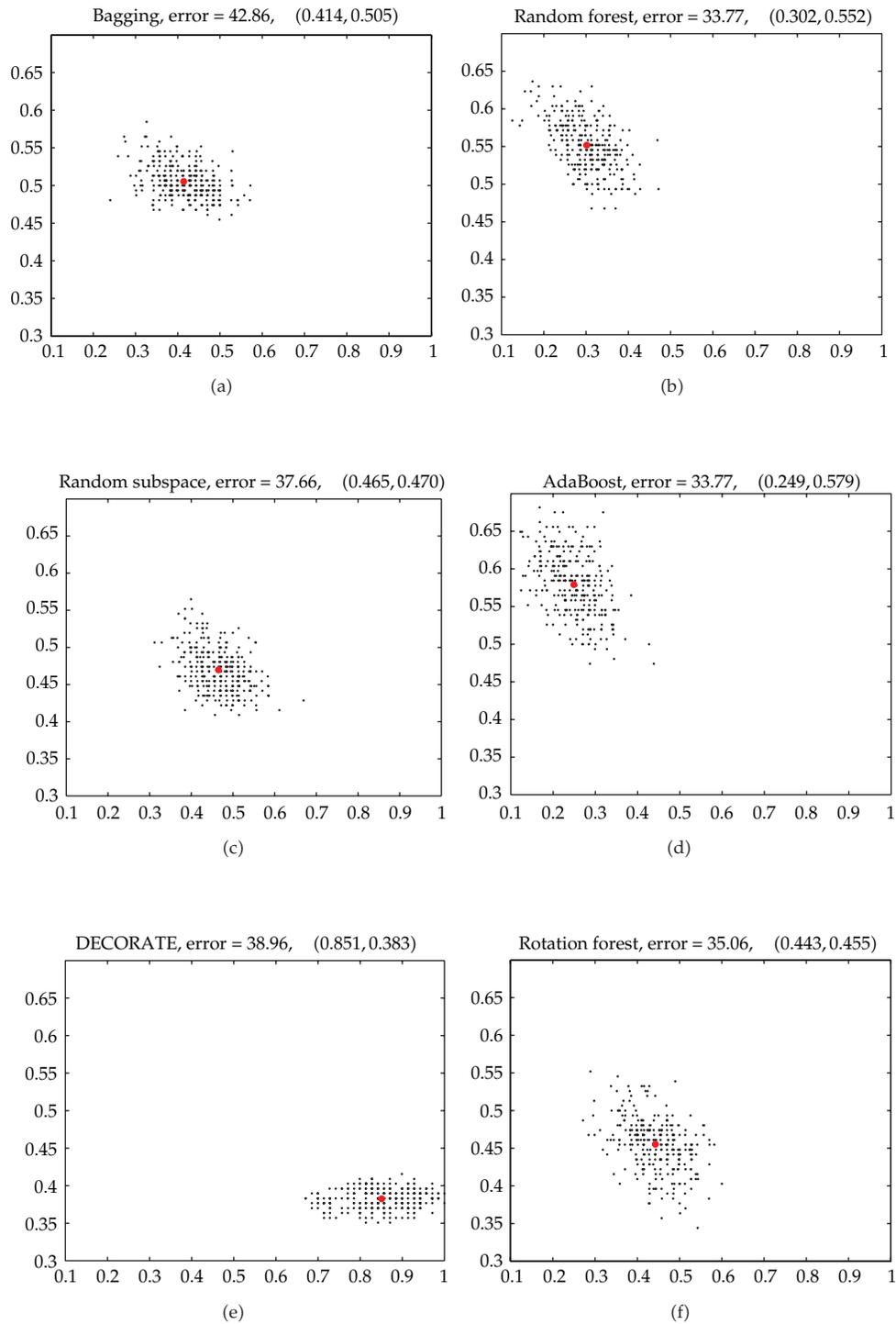


Figure 1: The kappa-error diagrams for the compared ensemble methods.

Thus, the final prediction error of rotation forest is a little higher than that of random forest and AdaBoost in this special situation.

Notice that here we just randomly selected 90% of the data to construct forests and utilized the remaining 10% to estimate the values of κ and mean error for each pair of base classifiers. The experiment was only carried out for one trial. If comparing the results obtained herein with those listed in Table 1, we should be cautious to draw conclusions since the errors for each algorithm as reported in Table 1 were averaged over ten trials of 10-fold cross-validation in order to eliminate some random factor that may affect the relative performance of our considered classification methods. However, our aim in this subsection was just to employ the kappa-error diagrams to study the working mechanism of the compared ensemble methods more clearly.

5. Conclusions

In this paper, we adopted some widely used classifier fusion methods to solve a hand gesture recognition problem. Since the data of this classification task are likely coming from a multi-normal distribution, the ensemble methods are found to be more appropriate to deal with this problem because the performance of them is much better than that of a single classification tree. Among the ensemble techniques, AdaBoost and rotation forest behave significantly better than their rivals and they achieve the lowest generalization error. Through investigating the bias-variance decompositions of error for the considered classification algorithms, the success of AdaBoost and rotation forest can be attributed to the fact that each of them simultaneously reduces the bias and variance terms of the SingleTree's error to a large extent. Rotation forest does a better job to reduce variance whereas AdaBoost has a small advantage on reducing bias. Furthermore, we made use of kappa-error diagrams to visualize how a classifier combination strategy attempts to reach a good tradeoff between diversity and accuracy in the process of constructing an ensemble classifier. The experimental results demonstrate that AdaBoost creates the most diverse base classifiers but with a little higher error. With respect to rotation forest, it is observed to generate very accurate base classifiers while the diversity between them is only medium.

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China (no. 61075006), the Tianyuan Special Funds of the National Natural Science Foundation of China (no. 11126277), the Fundamental Research Funds for the Central Universities of China, as well as the Research Fund for the Doctoral Program of Higher Education of China (no. 20100201120048). The authors would like to thank Gineke A. ten Holt for providing the hand gesture data. The authors are grateful to the reviewers as well as the editor for their valuable comments and suggestions which lead to a substantial improvement of the paper.

References

- [1] X. Zabulis, H. Baltzakis, and A. Argyros, "Vision-based hand gesture recognition for humancomputer interaction," in *The Universal Access Handbook, Human Factors and Ergonomics*, Lawrence Erlbaum Associates, 2009.
- [2] S. S. Ge, Y. Yang, and T. H. Lee, "Hand gesture recognition and tracking based on distributed locally linear embedding," *Image and Vision Computing*, vol. 26, no. 12, pp. 1607–1620, 2008.

- [3] J. F. Lichtenauer, E. A. Hendriks, and M. J. Reinders, "Sign language recognition by combining statistical DTW and independent classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 2040–2046, 2008.
- [4] B. C. Bedregal, A. C. R. Costa, and G. P. Dimuro, "Fuzzy rule-based hand gesture recognition," in *Artificial Intelligence in Theory and Practice*, pp. 285–294, Springer, 2009.
- [5] A. Just, R. Yann, and S. Marcel, "Hand posture classification and recognition using the modified census transform," in *Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, pp. 351–356, Southampton, UK, 2006.
- [6] T. B. Dinh, V. B. Dang, D. A. Duong, T. T. Nguyen, and D. D. Le, "Hand gesture classification using boosted cascade of classifiers," in *Proceedings of the International Conference on Research, Innovation and Vision for the Future*, pp. 139–144, Ho Chi Minh City, Vietnam, 2006.
- [7] T. K. Kim and R. Cipolla, "Gesture recognition under small sample size," in *Proceedings of the the 8th Asian Conference on Computer Vision*, Part I, LNCS 4843, pp. 335–344, 2007.
- [8] E. Stergiopoulou and N. Papamarkos, "Hand gesture recognition using a neural network shape fitting technique," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 8, pp. 1141–1158, 2009.
- [9] M. Flasiński and S. Myśliński, "On the use of graph parsing for recognition of isolated hand postures of Polish Sign Language," *Pattern Recognition*, vol. 43, no. 6, pp. 2249–2264, 2010.
- [10] T. Burger, O. Aran, A. Urankar, A. Caplier, and L. Akarun, "A Dempster-Shafer theory based combination of classifiers for hand gesture recognition," *Communications in Computer and Information Science*, vol. 21, pp. 137–150, 2008.
- [11] A. Just and S. Marcel, "A comparative study of two state-of-the-art sequence processing techniques for hand gesture recognition," *Computer Vision and Image Understanding*, vol. 113, no. 4, pp. 532–543, 2009.
- [12] O. Aran, T. Burger, A. Caplier, and L. Akarun, "A belief-based sequential fusion approach for fusing manual signs and non-manual signals," *Pattern Recognition*, vol. 42, no. 5, pp. 812–822, 2009.
- [13] N. C. Oza and K. Tumer, "Classifier ensembles: Select real-world applications," *Information Fusion*, vol. 9, no. 1, pp. 4–20, 2008.
- [14] Y. Zhang and Z. H. Zhou, "Cost-sensitive face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 10, Article ID 5342435, pp. 1758–1769, 2010.
- [15] R. Xia, C. Zong, and S. Li, "Ensemble of feature sets and classification algorithms for sentiment classification," *Information Sciences*, vol. 181, no. 6, pp. 1138–1152, 2011.
- [16] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.
- [17] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.
- [18] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [19] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, vol. 57 of *Monographs on Statistics and Applied Probability*, Chapman & Hall/CRC, London, UK, 1993.
- [20] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth Statistics/Probability Series, Chapman and Hall, New York, NY, USA, 1984.
- [21] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the 13th International Conference on Machine Learning*, pp. 148–156, Morgan Kaufmann, Bari, Italy, 1996.
- [22] 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, pp. 119–139, 1997.
- [23] J. J. Rodríguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: a new classifier ensemble method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619–1630, 2006.
- [24] A. Frank and A. Asuncion, *UCI Machine Learning Repository*, University of California, School of Information and Computer Science, Irvine, Calif, USA, 2010.
- [25] P. Melville and R. J. Mooney, "Creating diversity in ensembles using artificial data," *Information Fusion*, vol. 6, no. 1, pp. 99–111, 2005.
- [26] G. A. ten Holt, *Automatic sign language recognition inspired by human sign perception*, Ph.D. thesis, Delft University of Technology, 2010.
- [27] E. Bauer and R. Kohavi, "Empirical comparison of voting classification algorithms: bagging, boosting, and variants," *Machine Learning*, vol. 36, no. 1, pp. 105–139, 1999.
- [28] C. X. Zhang and J. S. Zhang, "RotBoost: a technique for combining Rotation Forest and AdaBoost," *Pattern Recognition Letters*, vol. 29, no. 10, pp. 1524–1536, 2008.
- [29] C. X. Zhang and J. S. Zhang, "A novel method for constructing ensemble classifiers," *Statistics and Computing*, vol. 19, no. 3, pp. 317–327, 2009.

- [30] R. Kohavi and D. Wolpert, "Bias plus variance decomposition for zero-one loss functions," in *Proceedings of the 13th International Conference on Machine Learning*, pp. 275–283, Morgan Kaufmann, Bari, Italy, 1996.
- [31] L. Breiman, "Arcing classifiers," *The Annals of Statistics*, vol. 26, no. 3, pp. 801–849, 1998.
- [32] G. I. Webb, "MultiBoosting: a technique for combining boosting and wagging," *Machine Learning*, vol. 40, no. 2, pp. 159–196, 2000.
- [33] G. M. James, "Variance and bias for general loss functions," *Machine Learning*, vol. 51, no. 2, pp. 115–135, 2003.
- [34] L. I. Kuncheva, *Combining Pattern Classifiers*, Wiley-Interscience, Hoboken, NJ, USA, 2004.
- [35] L. Rokach, "Taxonomy for characterizing ensemble methods in classification tasks: a review and annotated bibliography," *Computational Statistics & Data Analysis*, vol. 53, no. 12, pp. 4046–4072, 2009.
- [36] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, 2010.
- [37] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *Proceedings of the 14th International Conference on Machine Learning*, pp. 211–218, Morgan Kaufmann, San Mateo, Calif, USA, 1997.



Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

