Review of Classifier Combination Methods

Sergey Tulyakov[1], Stefan Jaeger[2], Venu Govindaraju[1], and David Doermann[2]

[1] Center for Unified Biometrics and Sensors, University at Buffalo, Amherst, NY 14228, USA `tulyakov@cedar.buffalo.edu,venu@cubs.buffalo.edu`
[2] Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA {`jaeger,doermann`}`@umiacs.umd.edu`

1 Introduction

The efforts to automate the combination of expert opinions have been studied extensively in the second half of the twentieth century[8]. These studies have covered diverse application areas: economic and military decisions, natural phenomena forecasts, technology applications. The combinations presented in these studies can be separated into mathematical and behavioral approaches[6]. The mathematical combinations try to construct models and derive combination rules using logic and statistics. The behavioral methods assume discussions between experts, and direct human involvement in the combination process. The mathematical approaches gained more attention with the development of computer expert systems. Expert opinions could be of different nature dependent on the considered applications: numbers, functions, etc. For example, the work of R. Clemen contains combinations of multiple types of data, and, in particular, considers combinations of experts' estimations of probability density functions [6].

The pattern classification field developed around the end of the twentieth century deals with the more specific problem of assigning input signals to two or more classes. The combined experts are classifiers and the result of the combination is also a classifier. The outputs of classifiers can be represented as vectors of numbers where the dimension of vectors is equal to the number of classes. As a result, the combination problem can be defined as a problem of finding the combination function accepting N-dimensional score vectors from M classifiers and outputting N final classification scores (Figure 1), where the function is optimal in some sense, e.g. minimizing the misclassification cost. In this chapter, we will deal exclusively with the mathematical methods for classifier combination from a pattern classification perspective.

In the last ten years, we have seen a major boost of publications in optical character recognition and biometrics. Pattern classification applications of these two fields include image classification, e.g character recognition and word

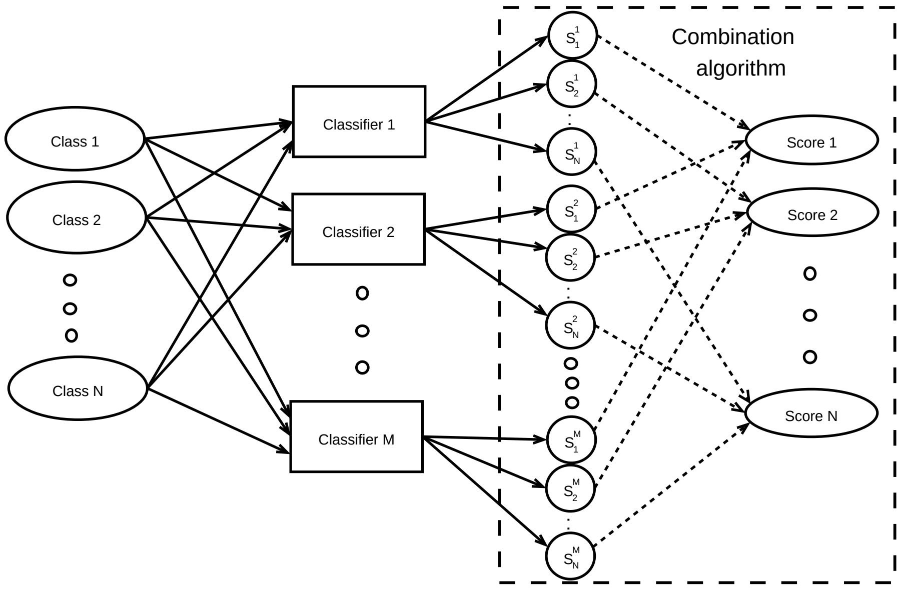

Fig. 1. Classifier combination takes a set of s_i^j - score for class i by classifier j and produces combination scores S_i for each class i.

recognition, speech recognition, person authentication by voice, face image, fingerprints or other biometric characteristics. As a result, these two fields are the most popular application targets for multiple classifier systems so far [61, 17, 49, 58].

In this chapter, we will present different categorizations of classifier combination methods. Our goal is to give a better understanding of the current problems in this field. We will also provide descriptions of major classifier combination methods and their applications in document analysis. We organized this chapter as follows: Section 2 introduces different categorizations of classifier combination methods. Section 3 discusses ensemble techniques, while Section 4 focuses on non-ensemble techniques. In Section 5, we address additional issues important to classifier combination, such as retraining.

2 Defining the Classifier Combination Problem

In order to provide a more complete description of the classifier combination field we attempt to categorize different types of classifier combinations in this section.

2.1 Score Combination Functions and Combination Decisions

Classifier combination techniques operate on the outputs of individual classifiers and usually fall into one of two categories. In the first approach the outputs are treated as inputs to a generic classifier, and the combination algorithm is created by training this, sometimes called 'secondary', classifier. For example, Dar-Shyang Lee [42] used a neural network to operate on the outputs of the individual classifiers and to produce the combined matching score. The advantage of using such a generic combinator is that it can learn the combination algorithm and can automatically account for the strengths and score ranges of the individual classifiers. In the second approach, a function or a rule combines the classifier scores in a predetermined manner.

The final goal of classifier combination is to create a classifier which operates on the same type of input as the base classifiers and separates the same types of classes. Using combination rules implies some final step of classification decision. If we denote the score assigned to class i by base classifier j as s_i^j, then the typical combination rule is some function f and the final combined score for class i is $S_i = f(\{s_i^j\}_{j=1,\dots,M})$. The sample is classified as belonging to class $\arg\max_i S_i$. Thus the combination rules can be viewed as a classifier operating on base classifiers' scores, involving some combination function f and the $\arg\max$ decision, showing that there is no real conceptual difference between the categories mentioned above.

Generic classifiers used for combinations do not have to be necessarily constructed following the above described scheme, but in practice we see this theme commonly used. For example, in multilayer perceptron classifiers the last layer has each node containing a final score for one class. These scores are then compared and the maximum is chosen. Similarly, k-nearest neighbor classifier can produce scores for all classes as ratios of the number of representatives of a particular class in a neighborhood to k. The class with the highest ratio is then assigned to a sample.

In summary, combination rules can be considered as a special type of classifier of particular $\arg\max f$ form. Combination functions f are usually simple functions, such as sum, weighted sum, max, etc. Generic classifiers such as neural networks and k-nearest neighbor, on the other hand, imply more complicated functions.

2.2 Combinations of Fixed Classifiers and Ensembles of Classifiers

One main categorization is based on whether the combination uses a fixed (usually less than 10) set of classifiers, as opposed to a large pool of classifiers (potentially infinite) from which one selects or generates new classifiers. The first type of combinations assumes classifiers are trained on different features or different sensor inputs. The advantage comes from the diversity of the classifiers' strengths on different input patterns. Each classifier might be an expert on certain types of input patterns. The second type of combinations assumes large number of classifiers, or ability to generate classifiers. In the second type of combination the large number of classifiers are usually obtained by selecting different subsets of training samples from one large training set, or by selecting different subsets of features from the set of all available features, and by training the classifiers with respect to selected training subset or subset of features.

2.3 Operating Level of Classifiers

Combination methods can also be grouped based on the level at which they operate. Combinations of the first type operate at the *feature* level. The features of each classifier are combined to form a joint feature vector and classification is subsequently performed in the new feature space. The advantage of this approach is that using the features from two sets at the same time can potentially provide additional information about the classes. For example, if two digit recognizers are combined in such a fashion, and one recognizer uses a feature indicating the enclosed area, and the other recognizer has a feature indicating the number of contours, then the combination of these two features in a single recognizer will allow class '0' to be easily separated from the other classes. Note that individually, the first recognizer might have difficulty separating '0' from '8', and the second recognizer might have difficulty separating '0' from '6' or '9'. However, the disadvantage of this approach is that the increased number of feature vectors will require a large training set and complex classification schemes. If the features used in the different classifiers are not related, then there is no reason for combination at the feature level.

Combinations can also operate at the decision or score level, that is they use outputs of the classifiers for combination. This is a popular approach because the knowledge of the internal structure of classifiers and their feature vectors is not needed. Though there is a possibility that representational information is lost during such combinations, this is usually compensated by the lower complexity of the combination method and superior training of the final system. In the subsequent sections we will only consider classifier combinations at the decision level.

The following is an example of the application where feature level combination can improve a classifier's performance. Bertolami and Bunke[3] compared the combinations at the feature and the decision levels for handwriting recognition. Their handwriting recognizer uses a sliding window to extract pixel and geometrical features for HMM matching. The combination at the feature level has a single HMM trained on the composite vector of these features. The combination at the decision level has two HMMs trained on separate pixel and geometrical feature vectors, and the recognition results, word matching scores, are combined together with the language model. The combination at the feature level seems to achieve better results, and authors explain its effectiveness by improved alignment of the HMM recognizer.

Note that combination on the feature level is not conceptually different from trying to incorporate different types of information into a single feature vector. For example, Favata[11] constructed handwritten word recognizers which utilized a character recognizer based on gradient, concavity and structural features. Thus feature level combination rather delegates the combination task to the base classifier (HMM in above example) instead of solving it.

2.4 Output Types of Combined Classifiers

Another way to categorize classifier combination is by the outputs of the classifiers used in the combination. Three types of classifier outputs are usually considered[61]:

- Type I (abstract level): This is the lowest level since a classifier provides the least amount of information on this level. Classifier output is merely a single class label or an unordered set of candidate classes.
- Type II (rank level): Classifier output on the rank level is an ordered sequence of candidate classes, the so-called n-best list. The candidate class at the first position is the most likely class, while the class positioned at the end of the list is the most unlikely. Note that there are no confidence values attached to the class labels on rank level. Only their position in the n-best list indicates their relative likelihood.
- Type III (measurement level): In addition to the ordered n-best lists of candidate classes on the rank level, classifier output on the measurement level has confidence values assigned to each entry of the n-best list. These confidences, or scores, can be arbitrary real numbers, depending on the classification architecture used. The measurement level contains therefore the most information among all three output levels.

In principle, a combination method can operate on any of these levels. For combinations based solely on label sets or rankings of class labels, i.e. output on abstract and rank level, several voting techniques have been proposed and experimentally investigated [23, 10, 33]. The advantage of classifier output on abstract and rank level is that different confidence characteristics have no negative impact on the final outcome, simply because confidence plays no role in the decision process. Nevertheless, the confidence of a classifier in a particular candidate class usually provides useful information that a simple class ranking cannot reflect. This suggests the use of combination methods that operate on the measurement level, and which can exploit the confidence assigned to each candidate class. Nowadays most classifiers do provide information on measurement level, so that applying combination schemes on the measurement level should be possible for most practical applications. On measurement level, however, we have to take into account that each classifier in a multiple classifier system may output quite different confidence values, with different ranges, scales, means etc. This may be a minor problem for classifier ensembles generated with Bagging and Boosting (see Section 3 since all classifiers in the ensemble are based on the same classification architecture, only their training sets differ. Each classifier will therefore provide similar output. However, for classifiers based on different classification architectures, this output will in general be different. Since different architectures lead more likely to complementary classifiers, which are especially promising for combination purposes, we need effective methods for making outputs of different classifiers comparable.

If the combination involves classifiers with different output types, the output is usually converted to any one of the above: to type I[61], to type II[23], or to type III[42]. Most existing classifier combination research deals with classifier outputs of type III (measurement level), among which we can find combinations with fixed structure, e.g. sum of scores [61, 34], or combinations that can be trained using available training samples (weighted sum, logistic regression[23], Dempster-Shafer rules [61], neural network[42], etc.).

2.5 Complexity Types of Classifier Combinations

In [51] we proposed a new framework for combining classifiers based on the structure of combination functions. Though we applied it only to biometric person authentication applications there, it can provide a useful insight into other applications as well.

As Figure 1 shows, the combination algorithm is a map

$$\{s_k^j\}_{j=1,\dots,M;k=1,\dots,N} \Rightarrow \{S_i\}_{i=1,\dots,N} \qquad (1)$$

of NM classifiers' scores into the N-dimensional final score space, where N is the number of classes. The complexity types define how such maps are constructed. Specifically, the combination type is determined by whether all classifiers' scores participate in the derivation of each final combined score, and whether only a single combination function is trained for all classes or there is an individual combination function for each class.

We separate combination algorithms into four different types depending on the number of classifier's scores they take into account and the number of combination functions required to be trained:

1. Low complexity combinations: $S_i = f(\{s_i^j\}_{j=1,\dots,M})$. Combinations of this type require only one combination function to be trained, and the combination function takes as input scores for one particular class.
2. Medium complexity I combinations: $S_i = f_i(\{s_i^j\}_{j=1,\dots,M})$. Combinations of this type have separate score combining functions for each class and each such function takes as input parameters only the scores related to its class.
3. Medium complexity II combinations:

$$S_i = f(\{s_i^j\}_{j=1,\dots,M}, \{s_k^j\}_{j=1,\dots,M;k=1,\dots,N,k\neq i}) \qquad (2)$$

This function takes as parameters not only the scores related to one class, but all output scores of classifiers. Combination scores for each class are calculated using the same function, but scores for class i are given a special place as parameters. Applying function f for different classes effectively means permutation of the function's parameters.

4. High complexity combinations: $S_i = f_i(\{s_k^j\}_{j=1,\dots,M;k=1,\dots,N})$. Functions calculating final scores are different for all classes, and they take as parameters all output base classifier scores.

In order to illustrate the different combination types we can use a matrix representation as shown in Figure 2. Each row corresponds to a set of scores output by a particular classifier, and each column has scores assigned by classifiers to a particular class.

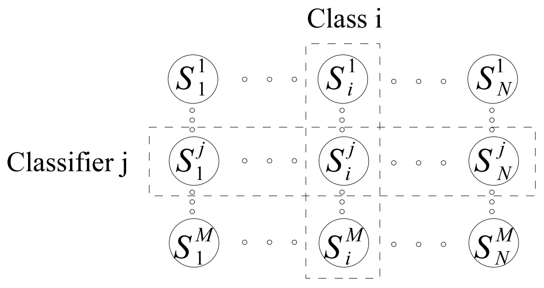

Fig. 2. Output classifier scores arranged in a matrix; s_i^j - score for class i by classifier j.

Figure 3 shows four complexity types of combinations using the score matrix. The combinations of low and medium I complexity types use only scores of one particular class to derive a final combined score for this class. Medium II and high complexity combinations use scores related to all classes to derive a final combined score of any single class. Low and medium II complexity combinations have a single combination function f used for all classes, and medium I and high complexity combinations might have different combination functions f_i for different classes i.

As an example, simple combination rules (sum, weighted sum, product, etc.) typically produce combinations of low complexity type. Combinations which try to find the separate combination function for each class [32, 12] are of medium I complexity type. The rank-based combination methods (e.g. Borda count in Section 4) represent the combinations of medium II complexity type, since calculating rank requires comparing the original score with other scores produced during the same identification trial. Behavior-knowledge spaces (BKS, see Section 4) are an example of high complexity combination type, since they are both rank-based and can have user-specific combination

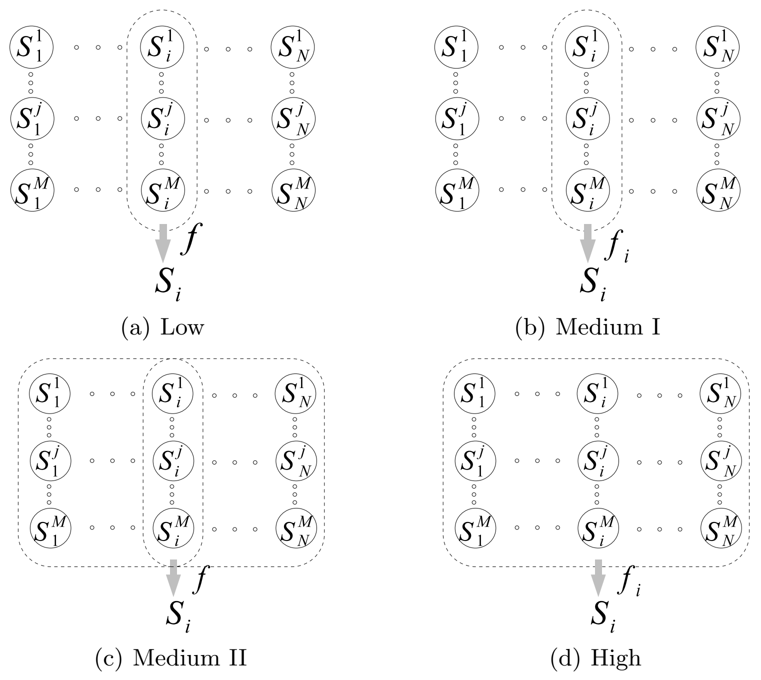

Fig. 3. The range of scores considered by each combination type and combination functions.

functions. One way to obtain combinations of different types is to use different score normalizations before combining normalized scores by a simple combination rule of low complexity. For example, by using class-specific Z-normalization or identification trial specific T-normalization [2], we are dealing respectively with medium I or medium II complexity combination types.

Higher complexity combinations can potentially produce better classification results since more information is used. On the other hand the availability of training samples will limit the types of possible combinations. Thus the choice of combination type in any particular application is a trade-off between classifying capabilities of combination functions and the availability of sufficient training samples. When the complexity is lowered it is important to see if any useful information is lost. If such loss happens, the combination algorithm should be modified to compensate for it.

Different generic classifiers such as neural networks, decision trees, etc., can be used for classifier combinations within each complexity class. However, the choice of the generic classifiers or combination functions is less important than the choice of the complexity type.

2.6 Classification and Combination

From Figure 1 and the discussion in Section 2.1 we can view the problem of combining classifiers as a classification problem in the score space $\{s_i^j\}_{j=1,\dots,M;i=1,\dots,N}$. Any generic pattern classification algorithm trained in this score space can act as a combination algorithm. Does it make sense to search for other, more specialized methods of combination? In other words, does classifier combination field has anything new to offer with respect to traditional pattern classification research?

One difference between the combination problem and the general pattern classification problem is that in the combination problem features (scores) have a specific meaning of being related to a particular class or being produced by a particular classifier. In the general pattern classification problem we do not assign such meaning to features. Thus intuitively we tend to construct combination algorithms which take such meaning of scores into consideration. The four combination complexity types presented in the previous section are based on this intuition, as they pay special attention to the scores s_i of class i while deriving a combined score S_i for this class.

The meaning of the scores, though, does not provide any theoretical basis for choosing a particular combination method, and in fact can lead to constructing suboptimal combination algorithms. For example, by constructing combinations of low and medium I complexity types we effectively disregard any interdependencies between scores related to different classes. As we showed in [51] and [52] such dependencies can provide useful information for the combination algorithm.

The following is a list of cases which might not be solved optimally in traditional pattern classification algorithms. The task of classifier combination can be defined as developing specialized combination algorithms for these cases.

1. The situation of having to deal with a large number of classes arises frequently in the pattern recognition field. For example, biometric person identification, speech and handwriting recognition are applications with very large number of classes. The number of samples of each class available for training can be small, such as in biometric applications where a single person template is enrolled into the database, or even zero for speech and handwriting recognition when the class is determined by the lexicon word.
2. The number of classifiers M is large. For example, taking multiple training sets in bagging and boosting techniques yields arbitrarily large number of classifiers. The usual method of combination in these cases is to use some a priori rule, e.g. sum rule.
3. Additional information about classifiers is available. For example, in the case of multimodal biometrics combination it is safe to assume that classifiers act independently. This might be used to better estimate the joint

score density of M classifiers as a product of M separately estimated score densities of each classifier.

4. Additional information about classes is available. Consider the problem of classifying word images into classes represented by a lexicon: The relation between classes can be expressed through classifier independent methods, for example, by using the string edit distance. Potentially classifier combination methods could benefit from such additional information.

The cases listed above present situations where generic pattern classification methods in score space are not sufficient or suboptimal. The first two cases describe scenarios where the feature space has very large dimensions. By adopting a combination of reduced complexity we are able to train combination algorithm and achieve performance improvement. If neither the number of classifiers nor the number of classes is large, the generic pattern classification algorithm operating in the score space can solve the combination problem.

When additional information besides training score vectors is available as in scenarios 3 and 4 it should be possible to improve on the generic classification algorithms which use only a sample of available score vectors for training, but no other information.

3 Classifier Ensembles

The focus of this chapter is to explore the combinations on a fixed set of classifiers. We assume that there are only few classifiers and we can collect some statistical data about these classifiers using a training set. The purpose of the combination algorithm is to learn the behavior of these classifiers and produce an efficient combination function.

In this section, however, we shortly address another approach to combination that includes methods trying not only to find the best combination algorithm, but also trying to find the best set of classifiers for the combination. This type of combination usually requires a method for generating a large number of classifiers. Few methods for generating classifiers for such combinations exist. One of the methods is based on bootstrapping the training set in order to obtain a multitude of subsets and train a classifier on each of these subsets. Another method is based on the random selection of the subsets of features from one large feature set and training classifiers on these feature subsets[41]. A third method applies different training conditions, e.g. choosing random initial weights for neural network training or choosing dimensions for decision trees [22]. The ultimate method for generating classifiers is a random separation of feature space into the regions related to particular classes [36].

Simplest methods of combination apply some fixed functions to the outputs of all the generated classifiers (majority voting, bagging [4]). More complex methods, such as boosting [45, 13], stack generalization [59], attempt to select only those classifiers which will contribute to the combination.

Although there is substantial research on the classifier ensembles, very few theoretical results exist. Most explanations use bias and variance framework which is presented below. But such approaches can only give asymptotic explanations of observed performance improvements. Ideally, the theoretical foundation for classifier ensembles should use statistical learning theory [54, 55]. But it seems that such work will be quite difficult. For example, it is noted in [46] that an unrestricted ensemble of classifiers has a higher complexity than individual combined classifiers. The same paper presents an interesting explanation of the performance improvements based on the classifier's margin - the statistical measure of the difference between scores given to correct and incorrect classification attempts. Another theoretical approach to the classifier ensemble problem was developed by Kleinberg in the theory of stochastic discrimination[35, 36]. This approach considers very general type of classifiers (which are determined by the regions in the feature space) and outlines criteria on how these classifiers should participate in the combination.

3.1 Reductions of Trained Classifier Variances

One way to explain the improvements observed in ensemble combination methods (bagging, boosting) is to decompose the added error of the classifiers into bias and variance components[37, 50, 4]. There are few definitions of such decompositions[13]. Bias generally shows the difference between optimal Bayesian classification and average of trained classifiers, where average means real averaging of scores or voting and average is taken over all possible trained classifiers. The variance shows the difference between a typical trained classifier and an average one.

The framework of Tumer and Ghosh[53] associates trained classifiers with the approximated feature vector densities of each class. This framework has been used in many papers on classifier combination recently[40, 38, 15, 16]. In this framework, trained classifiers provide approximations to the true posterior class probabilities or to the true class densities:

$$f_i^m(x) = p_i(x) + \epsilon_i^m(x) \tag{3}$$

where i is the class index and m is the index of a trained classifier. For a fixed point x the error term can be represented as a random variable where randomness is determined by the random choice of the classifier or used training set. By representing it as a sum of mean β and zero-mean random variable η we get

$$\epsilon_i^m(x) = \beta_i(x) + \eta_i^m(x) \tag{4}$$

For simplicity, assume that the considered classifiers are unbiased, that is $\beta_i(x) = 0$ for any x, i. If point x is located on the decision boundary between classes i and j then the added error of the classifier is proportional to the sum of the variances of η_i and η_j:

$$E^m_{add} \sim \sigma^2_{\eta^m_i} + \sigma^2_{\eta^m_j} \tag{5}$$

If we average M such trained classifiers and if the error random variables η^m_i are independent and identically distributed as η_i, then we would expect the added error to be reduced M times:

$$E^{ave}_{add} \sim \sigma^2_{\eta^{ave}_i} + \sigma^2_{\eta^{ave}_j} = \frac{\sigma^2_{\eta_i} + \sigma^2_{\eta_j}}{M} \tag{6}$$

The application of the described theory is very limited in practice since too many assumptions about classifiers are required. Kuncheva[38] even compiles a list of used assumptions. Besides independence assumption of errors, we need to hypothesize about error distributions, that is the distributions of the random variable η_i. The tricky part is that η_i is the difference between true distribution $p_i(x)$ and our best guess about this distribution. If we knew what the difference is, we would have been able to improve our guess in the first place. Although there is some research [40, 1] into trying to make assumptions about these estimation error distributions and seeing which combination rule is better for a particular hypothesized distribution, the results are not proven in practice.

3.2 Bagging

Researchers have very often concentrated on improving single-classifier systems mainly because of their lack in sufficient resources for simultaneously developing several different classifiers. A simple method for generating multiple classifiers in those cases is to run several training sessions with the same single-classifier system and different subsets of the training set, or slightly modified classifier parameters. Each training session then creates an individual classifier. The first more systematic approach to this idea was proposed in [4] and became popular under the name “Bagging.”This method draws the training sets with replacement from the original training set, each set resulting in a slightly different classifier after training. The technique used for generating the individual training sets is also known as bootstrap technique and aims at reducing the error of statistical estimators. In practice, bagging has shown good results. However, the performance gains are usually small when bagging is applied to weak classifiers. In these cases, another intensively investigated technique for generating multiple classifiers is more suitable: Boosting.

3.3 Boosting

Boosting has its root in a theoretical framework for studying machine learning, and deals with the question whether an almost randomly guessing classifier can be boosted into an arbitrarily accurate learning algorithm. Boosting attaches a weight to each instance in the training set [14, 13, 20]. The weights are updated after each training cycle according to the performance of the classifier on the corresponding training samples. Initially, all weights are set equally, but on each round, the weights of incorrectly classified samples are increased so that the classifier is forced to focus on the hard examples in the training set [14].

A very popular type of boosting is AdaBoost (Adaptive Boosting), which was introduced by Freund and Schapire in 1995 to expand the boosting approach introduced by Schapire. The AdaBoost algorithm generates a set of classifiers and votes them. It changes the weights of the training samples based on classifiers previously built (trials). The goal is to force the final classifiers to minimize expected error over different input distributions. The final classifier is formed using a weighted voting scheme. Details of AdaBoost, in particular the AdaBoost variant called AdaBoost.M1, can be found in [13].

Boosting has been successfully applied to a wide range of applications. Nevertheless, we will not go more into the details of boosting and other ensemble combinations in this paper. The reason is that the focus of classifier ensembles techniques lies more on the generation of classifiers and less on their actual combination.

4 Non-Ensemble Combinations

Non-ensemble combinations typically use a smaller number of classifiers than ensemble-based classifier systems. Instead of combining a large number of automatically generated homogeneous classifiers, non-ensemble classifiers try to combine heterogeneous classifiers complementing each other. The advantage of complementary classifiers is that each classifier can concentrate on its own small subproblem instead of trying to cope with the classification problem as a whole, which may be too hard for a single classifier. Ideally, the expertise of the specialized classifiers do not overlap. There are several ways to generate heterogeneous classifiers. The perhaps easiest method is to train the same classifier with different feature sets and/or different training parameters. Another possibility is to use multiple classification architectures, which produce different decision boundaries for the same feature set. However, this is not only more expensive in the sense that it requires the development of independent classifiers, but it also raises the question of how to combine the output provided by multiple classifiers.

Many combination schemes have been proposed in the literature. As we have already discussed, they range from simple schemes to relatively complex combination strategies. This large number of proposed techniques shows the uncertainty researchers still have in this field. Up till now, researchers have not been able to show the general superiority of a particular combination scheme, neither theoretically nor empirically. Though several researchers have come up with theoretical explanations supporting one or more of the proposed schemes, a commonly accepted theoretical framework for classifier combination is still missing.

4.1 Elementary Combination Schemes on Rank Level

The probably simplest way of combining classifiers are voting techniques on rank level. Voting techniques do not consider the confidence values that may have been attached to each class by the individual classifiers. This simplification allows easy integration of all different kinds of classifier architectures.

Majority voting

A straightforward voting technique is majority voting. It considers only the most likely class provided by each classifier and chooses the most frequent class label among this crisp output set. In order to alleviate the problem of ties, the number of classifiers used for voting is usually odd. A trainable variant of majority voting is weighted majority voting, which multiplies each vote by a weight before the actual voting. The weight for each classifier can be obtained; e.g., by estimating the classifiers' accuracies on a validation set. Another voting technique that takes the entire n-best list of a classifier into account, and not only the crisp 1-best candidate class, is Borda count.

Borda count

Borda count is a voting technique on rank level [10]. For every class, Borda count adds the ranks in the n-best lists of each classifier, with the first entry in the n-best list; i.e., the most likely class label, contributing the highest rank number and the last entry having the lowest rank number. The final output label for a given test pattern X is the class with highest overall rank sum. In mathematical terms, this reads as follows: Let N be the number of classifiers, and r_i^j the rank of class i in the n-best list of the j-th classifier. The overall rank r_i of class i is thus given by

$$r_i = \sum_{j=1}^{N} r_i^j \tag{7}$$

The test pattern X is assigned the class i with the maximum overall rank count r_i.

Borda count is very simple to compute and requires no training. Similar to majority vote, there is a trainable variant that associates weights to the ranks of individual classifiers. The overall rank count for class i then computes as

$$r_i = \sum_{j=1}^{N} w_j r_i^j \tag{8}$$

Again, the weights can be the performance of each individual classifier measured on a training or validation set.

While voting techniques provide a fast and easy method for improving classification rates, they nevertheless leave the impression of not realizing the full potential of classifier combination by throwing away valuable information on measurement level. This has lead scientists to experiment with elementary combination schemes on this level as well.

4.2 Elementary Combination Schemes on Measurement Level

Elementary combination schemes on measurement level apply simple rules for combination, such as sum-rule, product-rule, and max-rule. Sum-rule simply adds the score provided by each classifier of a classifier ensemble for every class, and assigns the class label with the maximum score to the given input pattern. Analogously, product-rule multiplies the score for every class and then outputs the class with the maximum score. Interesting theoretical results, including error estimations, have been derived for those simple combination schemes. For instance, Kittler et al. showed that sum-rule is less sensitive to noise than other rules [34]. Despite their simplicity, simple combination schemes have resulted in high recognition rates, and it is by no means obvious that more complex methods are superior to simpler ones, such as sum-rule.

The main problem with elementary combination schemes is the incompatibility of confidence values. As discussed in Subsection 2.4, the output of classifiers can be of different type. Even for classifier output on measurement level (Type III), the output can be of different nature; e.g., similarity measures, likelihoods in the statistical sense, or distances to hyperplanes. In fact, this is why many researchers prefer using the name "score," instead of the names "confidence" or "likelihood," for the values assigned to each class on measurement level. This general name should emphasize the fact that those values are not the correct a posteriori class probabilities and have, in general, neither the same range and scale nor the same distribution. In other words, scores need to be normalized before they can be combined in a meaningful way with elementary combination schemes.

Similar to the situation for combination schemes, there is no commonly accepted method for normalizing scores of different classifiers. A couple of normalization techniques have been proposed. According to Jain et al., a good normalization scheme must be robust and efficient [25, 31]. In this context, robustness refers to insensitivity to score outliers and efficiency refers to the proximity of the normalized values to the optimal values when the distribution of scores is known.

The perhaps easiest normalization technique is the min-max normalization. For a given set of matching scores $\{s_k\}, k = 1, 2, \ldots, n$, the min-max normalized scores $\{s'_k\}$ are given by

$$s'_k = \frac{s_k - min}{max - min}, \tag{9}$$

where max and min are the maximum and minimum estimated from the given set of matching scores $\{s_k\}$, respectively. This simple type of normalization retains the original distribution of scores except for a scaling factor, transforming all the scores into a common range $[0\,;1]$. The obvious disadvantage of min-max normalization is its sensitivity to outliers in the data used for estimation of min and max. Another simple normalization method, which is, however, sensitive to outliers as well, is the so-called z-score. The z-score computes the arithmetic mean μ and the standard deviation σ on the set of scores $\{s_k\}$, and normalizes each score with

$$s'_k = \frac{s_k - \mu}{\sigma} \tag{10}$$

It is biased towards Gaussian distributions and does not guarantee a common numerical range for the normalized scores. A normalization scheme that is insensitive to outliers, but also does not guarantee a common numerical range, is MAD. This stands for "median absolute deviation," and is defined as follows:

$$s'_k = \frac{s_k - median}{MAD}, \tag{11}$$

with $MAD = median(|s_k - median|)$. Note that the median makes this normalization robust against extreme points. However, MAD normalization does a poor job in retaining the distribution of the original scores. In [31], Jain et al. list two more normalization methods, namely a technique based on a double sigmoid function suggested by Cappelli et al. [5], and a technique called "tanh normalization" proposed by Hampel et al. [21]. The latter technique is both robust and efficient.

Instead of going into the details of these two normalization methods, we suggest an alternative normalization technique that we have successfully applied to document processing applications, in particular handwriting recognition and script identification. This technique was first proposed in [56] and [57]. Its basic idea is to perform a warping on the set of scores, aligning the nominal progress of score values with the progress in recognition rate. In mathematical terms, we can state this as follows:

$$s'_k = \frac{\sum\limits_{i=0}^{k} n_{correct}(s_i)}{N} \tag{12}$$

The help function $n_{correct}(s_i)$ computes the number of patterns that were correctly classified with the original score s'_k on an evaluation set with N patterns. The new normalized scores s'_k thus describe a monotonously increasing partial sum, with the increments depending on the progress in recognition rate. We can easily see that the normalized scores fall all into the same numerical range $[0\,;1]$. In addition, the normalized scores also show robustness against outliers because the partial sums are computed over a range of original scores, thus averaging the effect of outliers. Using the normalization scheme in (12), we were able to clearly improve the recognition rate of a combined on-line/off-line handwriting recognizer in [56, 57]. Combination of off-line and on-line handwriting recognition is an especially fruitful application domain of classifier combination. It allows combination of the advantages of off-line recognition with the benefits of on-line recognition, namely the independence from stroke order and stroke number in off-line data, such as scanned handwritten documents, and the useful dynamic information contained in on-line data, such as data captured by a Tablet PC or graphic tablet. Especially on-line handwriting recognition can benefit from a combined recognition approach because off-line images can easily be generated from on-line data.

In a later work, we elaborated the idea into an information-theoretical approach to sensor fusion, identifying the partial sum in (12) with an exponential distribution [28, 29, 30]. In this information-theoretical context, the normalized scores read as follows:

$$s'_k = -E * \ln\left(1 - p(s_k)\right) \tag{13}$$

The function $p(s_k)$ is an exponential distribution with an expectation value E that also appears as a scaler upfront the logarithmic expression. The function $p(s_k)$ thus describes an exponential distribution defining the partial sums in (12). Note that the new normalized scores, which we refer to as "informational confidence," are information defined in the Shannon sense as the negative logarithm of a probability [48]. With the normalized scores being information, sum-rule now becomes the natural combination scheme. For more details on this information-theoretical technique, including practical experiments, we refer readers to the references [28, 29, 30] and to another chapter in this book.

4.3 Dempster-Shafer Theory of Evidence

Among the first more complex approaches for classifier combination was the Dempster-Shafer theory of evidence [9, 47]. As its name already suggests, this theory was developed by Arthur P. Dempster and Glenn Shafer in the sixties and seventies. It was first adopted by researchers in Artificial Intelligence in order to process probabilities in expert systems, but has soon been adopted for other application areas, such as sensor fusion and classifier combination. Dempster-Shafer theory is a generalization of the Bayesian theory of probability and differs in several aspects: First, Dempster-Shafer's theory introduces degrees of belief that do not necessarily meet the mathematical properties of probabilities. Second, it assigns probabilities to sets of possible outcomes rather than single events only. Third, it considers probability intervals that contain the precise probability for sets of possible outcomes. The two main ideas of Dempster-Shafer's theory, as they are usually presented in textbooks, are to obtain degrees of belief for one question from subjective probabilities for a related question, and Dempster's rule for combining such degrees of belief when they are based on independent items of evidence [Glenn Shafer]. Dempster's rule of combination is a generalization of Bayes' rule. It operates on masses assigning probabilities to sets of outcomes. For two sets of masses m_1 and m_2, Dempster's rule defines the joint mass $m_{1,2}(X)$ for an outcome set X as follows:

$$m_{1,2}(X) = \begin{cases} 0 & \text{if} \quad X = \emptyset \\ \frac{1}{1-K} \sum\limits_{A_i \cap B_j = X} m_1(A_i) m_2(B_j) & \text{if} \quad X \neq \emptyset \end{cases} \tag{14}$$

where

$$K = \sum_{A_i \cap B_j = \emptyset} m_1(A_i) m_2(B_j) \tag{15}$$

The Dempster-Shafer approach has produced good results in document processing and its still used today, notwithstanding its higher complexity compared to other approaches [43].

4.4 Behavior Knowledge Space

An equally complex, but perhaps more popular approach is the Behavior-Knowledge Space (BKS) method introduced in [24]. The BKS method is a trainable combination scheme on abstract level, requiring neither measurements nor ordered sets of candidate classes. It tries to estimate the a posteriori probabilities by computing the frequency of each class for every possible set of classifier decisions, based on a given training set. The result is a lookup table that associates the final classification result with each combination of classifier outputs; i.e., each combination of outputs in the lookup table is represented by its most often encountered class label. Given a specific classifier decision $S_1, \ldots, S_N$ from N individual classifiers, the a posteriori probability $\hat{P}(c_i|S_1, \ldots, S_N)$ of class c_i is estimated as follows

$$\hat{P}(c_i|S_1, \ldots, S_N) = \frac{N(c_i|S_1, \ldots, S_N)}{\sum\limits_i N(c_i|S_1, \ldots, S_N)} \tag{16}$$

where $N(c_i|S_1, \ldots, S_N)$ counts the frequency of class c_i for each possible combination of crisp classifier outputs.

In order to provide reasonable performance, the BKS method needs to be trained with large datasets so that meaningful statistics can be computed for each combination in the lookup table. If k is the number of classes and N is the number of combined classifiers, then BKS requires estimates of k^{N+1} a posteriori probabilities. This can pose problems when this number is high but the available set of training patterns is only small.

In addition to the methods presented in this section, several other techniques have been proposed in the recent past [22, 26]. They have gained some importance, but are out of the scope of this paper.

5 Additional Issues in Classifier Combinations

5.1 The Retraining Effect

The combined classifiers usually produce complementary results and combination algorithms utilize this property. But there is another reason why combinations might produce the performance improvement. The improvement might be the result of the combination algorithm retraining the imperfect matching score outputs of combined classifiers on new training data.

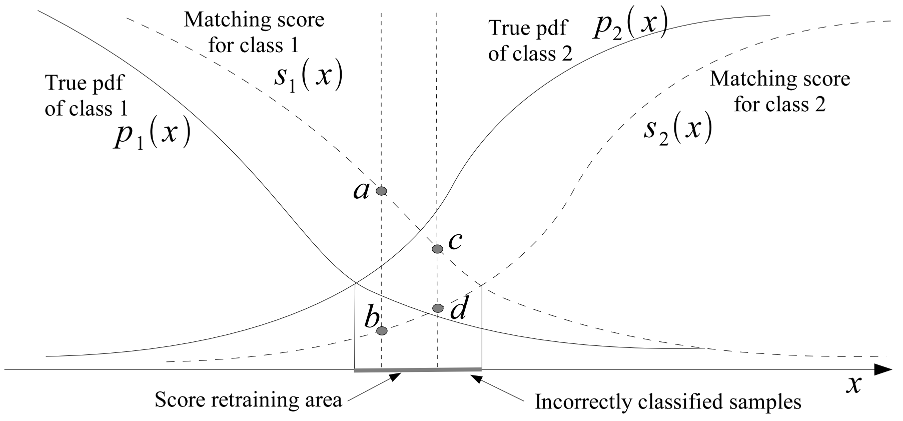

Fig. 4. Using additional training samples in the region of incorrectly classified samples might result in the correction of classification errors

Suppose we have two classes with equal prior probability and equal misclassification cost as in Figure 4. The optimal classification algorithm should make a decision based on whether the probability density function (pdf) of one class is bigger or less than that of the other class: $p_1(x) <> p_2(x)$. Suppose we have one classifier and it outputs a matching scores $s_1(x)$ and $s_2(x)$ for two classes approximating the pdfs of the corresponding classes. The decision of this classifier will be based on its scores: $s_1(x) <> s_2(x)$. The difference between these two decisions produces an area in which classification decisions are different from ideal optimal decisions ("Incorrectly classified samples" area of Figure 4). The classification performance can be improved if the decisions in this area are reversed.

The range of matching scores can be separated into regions of the type $c \leq s_1(x) \leq a$, or $b \leq s_2(x) \leq d$ or the intersections of those ("Score retraining area" of Figure 4). By using an additional set of training samples, the postprocessing algorithm can calculate the frequencies of training samples of each class belonging to these regions, say F_1 and F_2. If the number of training samples falling in these regions is large, then the frequencies will approximate the original class pdfs: $F_1 \sim p_1(x)$ and $F_2 \sim p_2(x)$. Consequently, the scores $s_1(x)$ and $s_2(x)$ could be replaced by new scores corresponding to frequencies F_1 and F_2, respectively. It is quite possible that the classification based on new scores will be correct in previously incorrectly classified regions.

This technique was implemented in [27] for the problem of classifying handwritten digits. The algorithm was able to improve the correctness of a classifier with 97% recognition rate by around 0.5% and of a classifier with 90% recognition rate by around 1.5%. Approximately 30,000 digit images were used for retraining and no information about the original training sets was provided.

This type of score postprocessing can be considered as a classifier combination function f which works only with one classifier and transforms the scores of this classifier s_1, s_2 into the combined scores S_1, S_2:

$$c \le s_1 \le a \quad \& \quad b \le s_2 \le d \quad \Longrightarrow \quad (S_1, S_2) = f(s_1, s_2) = (F_1, F_2)$$

This combination function is trained on newly supplied training samples and achieves improvement only due to the extraneous training data. Similarly, if we consider traditional combinations utilizing two or more classifiers, then some of the performance improvement might be attributed to using additional training data.

As far as we know, the retraining effect on classifier combinations has not been investigated so far. In particular, it would be interesting to know for each practical application, what part of the improvement is due to the retraining.

5.2 Locality Based Combination Methods

The retraining algorithm of the previous section implied the ability to separate training samples based on their class matching scores. The score for the test sample can be corrected if we are able to find a reference subset of training samples having similar performance. There exist few attempts to explore similar ideas in classifier combinations.

Behavior-knowledge spaces [24] can be considered as one of such locality based methods. The reference set is the subset of the training samples having same ranks assigned by all classifiers as a test sample. The choice of such reference sets might seem not very efficient - their number is large, and even single rank change places training sample in a different reference set. But experimental results seem to confirm the validity of such reference sets. Though the full technique is applicable only in cases with small number of classes and classifiers (two classifiers of ten digit classes in [24]), it can be extended to other cases by grouping together relevant subsets, e.g. subsets having same ranks for first few classes. Note, that rank information implicitly accounts for the dependence between scores assigned to different classes. By incorporating this information into the combination algorithm we effectively perform medium II or high complexity combinations.

Another type of forming reference sets is to use some additional information about classes. In the method of local accuracy estimates [60] the reference set for each test sample is found by applying the k-nearest neighbor algorithm to the original feature space. Subsequently, the strength of each classifier is determined on this reference set, and the classification decision is made by the classifier with highest performance. Clearly, the method could be extended to not only selecting the best performing classifier for the current test sample, but to combine the scores of classifiers according to their strengths. In any case, the important information about the test sample and its relationships with training samples is supplied by the k-nearest neighbor method. In a sense, the k-nearest neighbor method can be viewed as an additional classifier, which is combined with other classifiers. Similar approaches are also investigated in [18, 39].

Opposite to the presentation of the retraining effect in the previous subsection, the methods of finding reference sets discussed here do not rely exclusively on the matching scores of classifiers. Thus, by using additional information these methods can potentially perform better than the generic classifier operating in the score space. But, due to the similarity with retraining methods, it might be that a significant part of the performance improvement is caused by the retraining effect.

5.3 Locality Based Methods with Large Number of Classes

The interesting problem of finding the reference set for the test sample arises in the situations with large or variable number of classes. As we discussed in Subsection 2.6, such problems are not easily solvable with traditional pattern classification algorithms. For example, the recognition of handwritten words might include a large word lexicon, and the training data might simply not include all these words. In biometric person authentication only a small number of training samples, e.g. a single sample, per person is available. How can a reference set of training data be found in these cases?

The string edit distance might provide a valuable neighborhood information for the handwritten word application in order to find a reference set. In a more complicated approach we introduced the concept of local lexicon density in [19]. The lexicon density is determined not only by the lexicon itself, but also by the strength of the word recognizer using this lexicon. In addition to finding a reference set, the presented algorithm could be used to estimate the strength of each recognizer in the neighborhood of the test sample. Though the method of estimating lexicon density seems to have good performance, it has not been applied to problem of combining word recognizers yet.

The problem of finding a reference set for biometric person authentication has been investigated before in speaker identification research [7, 44]. The set of persons having similar biometrics to the queried person (cohort) is found using training samples of all enrolled persons. During testing, a matching score for the query person is judged against the matching scores of the cohort persons. These research was mostly used for making decisions on the matching scores, and not in combining different matchers. Some research has appeared recently [32, 12] trying to find user specific combinations of biometric matchers. But these methods use locality information only implicitly by estimating performance on the whole set of enrolled persons, and not on the cohort set. As far as we know, the cohort method has not been applied to classifier combinations so far.

The local neighborhood in cohorts is determined by the matching distances among enrolled templates. These distances can be used in constructing so called 'background models' [44]. In our research we used the set of matching scores between input template and enrolled templates to construct so called 'identification models' [52]. Both models can be used for finding reference sets in classifier combination algorithms. In addition, both models can be user-generic or user-specific, and both models can be used in a single combination method.

6 Conclusion

This chapter presented an overview of classifier combination methods. We categorized these methods according to complexity type, output type, ensemble vs non-ensemble combinations, etc. We also tried to define the classifier combination field by specifying cases, e.g. the presence of a large number of classes, which can not be readily solved by traditional pattern classification algorithms. We briefly presented main research directions in ensemble classifier combinations (Section 3) and non-ensemble classifier combinations (Section 4). In the last section, we discussed the retraining effect and issues of locality in classifier combinations. This section also presented some potential new research topics.

Overall, our goal was to show the current state of the art in classifier combination. Though significant progress has been made so far in devising new combination methods, the research is still mostly limited to the low complexity type of combinations. Exploring other complexity types of combinations and understanding locality properties of classifiers can be a fertile ground for future research.

References

1. F. M. Alkoot and J. Kittler. Experimental evaluation of expert fusion strategies. *Pattern Recognition Letters*, 20(11-13):1361–1369, 1999.
2. Roland Auckenthaler, Michael Carey, and Harvey Lloyd-Thomas. Score normalization for text-independent speaker verification systems. *Digital Signal Processing*, 10(1-3):42–54, 2000.
3. R. Bertolami and H. Bunke. Early feature stream integration versus decision level combination in a multiple classifier system for text line recognition. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 845–848, 2006.

4. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
5. R. Cappelli, D. Maio, and D. Maltoni. Combining Fingerprint Classifiers. In *First Int. Workshop on Multiple Classifier Systems*, pages 351–361, 2000.
6. R. Clemen and R. Winkler. Combining probability distributions from experts in risk analysis. *Risk Analysis*, 19:187–203, 1999.
7. J.M. Colombi, J.S. Reider, and J.P. Campbell. Allowing good impostors to test. In *Signals, Systems & Computers, 1997. Conference Record of the Thirty-First Asilomar Conference on*, volume 1, pages 296–300 vol.1, 1997.
8. R. Cooke. *Experts in Uncertainty: Opinion and Subjective Probability in Science.* Oxford University Press, 1991.
9. A. P. Dempster. A Generalization of Bayesian Inference. *Journal of the Royal Statistical Society*, 30:205–247, 1968.
10. M. Van Erp, L. G. Vuurpijl, and L. Schomaker. An Overview and Comparison of Voting Methods for Pattern Recognition. In *Proc. of the 8th International Workshop on Frontiers in Handwriting Recognition (IWFHR-8)*, pages 195–200, Niagara-on-the-Lake, Canada, 2002.
11. J.T. Favata. Character model word recognition. In *Fifth International Workshop on Frontiers in Handwriting Recognition*, pages 437–440, Essex, England, 1996.
12. Julian Fierrez-Aguilar, Daniel Garcia-Romero, Javier Ortega-Garcia, and Joaquin Gonzalez-Rodriguez. Bayesian adaptation for user-dependent multi-modal biometric authentication. *Pattern Recognition*, 38(8):1317–1319, 2005.
13. Y. Freund and R. Schapire. Experiments with a New Boosting Algorithm. In *Proc. of 13th Int. Conf. on Machine Learning*, pages 148–156, Bari, Italy, 1996.
14. Y. Freund and R. Schapire. A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.
15. G. Fumera and F. Roli. Performance analysis and comparison of linear combiners for classifier fusion. In *SSPR/SPR*, pages 424–432, 2002.
16. G. Fumera and F. Roli. Analysis of error-reject trade-off in linearly combined multiple classifiers. *Pattern Recognition*, 37(6):1245–1265, 2004.
17. P. Gader, M. Mohamed, and J. Keller. Fusion of Handwritten Word Classifiers. *Pattern Recognition Letters*, 17:577–584, 1996.
18. Giorgio Giacinto and Fabio Roli. Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition*, 34(9):1879–1881, 2001.
19. V. Govindaraju, P. Slavik, and Hanhong Xue. Use of lexicon density in evaluating word recognizers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(6):789–800, 2002.
20. S. Guenter and H. Bunke. New Boosting Algorithms for Classification Problems with Large Number of Classes Applied to a Handwritten Word Recognition Task. In *4th International Workshop on Multiple Classifier Systems (MCS)*, pages 326–335, Guildford, UK, 2003. Lecture Notes in Computer Science, Springer-Verlag.
21. F. R. Hampel, P. J. Rousseeuw, E.M. Ronchetti, and W.A. Stahel. *Robust Statistics: The Approach Based on Influence Functions.* Wiley, New York, 1986.
22. T. K. Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832–844, 1998.
23. T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994.

24. Y. Huang and C. Suen. A Method of Combining Multiple Experts for Recognition of Unconstrained Handwritten Numerals. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 17(1):90–94, 1995.
25. P. J. Huber. *Robust Statistics.* Wiley, New York, 1981.
26. K. Ianakiev and V. Govindaraju. Architecture for Classifier Combination Using Entropy Measures. In *1st International Workshop on Multiple Classifier Systems (MCS)*, pages 340–350, Cagliari, Italy, 2000. Lecture Notes in Computer Science, Springer-Verlag.
27. K. Ianakiev and V. Govindaraju. Deriving pseudo-probabilities of correctness given scores. In D. Chen and X. Cheng, editors, *Pattern Recognition and String Matching,*. Kluwer Publishers, 2002.
28. S. Jaeger. Informational Classifier Fusion. In *Proc. of the 17th Int. Conf. on Pattern Recognition*, pages 216–219, Cambridge, UK, 2004.
29. S. Jaeger. Using Informational Confidence Values for Classifier Combination: An Experiment with Combined On-Line/Off-Line Japanese Character Recognition. In *Proc. of the 9th Int. Workshop on Frontiers in Handwriting Recognition*, pages 87–92, Tokyo, Japan, 2004.
30. S. Jaeger, H. Ma, and D. Doermann. Identifying Script on Word-Level with Informational Confidence. In *Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 416–420, Seoul, Korea, 2005.
31. A. Jain, K. Nandakumar, and A. Ross. Score Normalization in Multimodal Biometric Systems. *Pattern Recognition*, 38(12):2270–2285, 2005.
32. A.K. Jain and A. Ross. Learning user-specific parameters in a multibiometric system. In *International Conference on Image Processing. 2002*, volume 1, pages I–57–I–60 vol.1, 2002.
33. H.-J. Kang and J.H. Kim. A Probabilistic Framework for Combining Multiple Classifiers at Abstract Level. In *Fourth International Conference on Document Analysis and Recognition (ICDAR)*, pages 870–874, Ulm, Germany, 1997.
34. J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 226–239, March 1998.
35. E. M. Kleinberg. Stochastic discrimination. *Annals of Mathematics and Artificial Intelligence*, 1, 1990.
36. E. M. Kleinberg. On the algorithmic implementation of stochastic discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):473–490, 2000.
37. E. Kong and T. Dietterich. Error-correcting output coding corrects bias and variance. In *12th International Conference on Machine Learning*, pages 313–321, 1995.
38. L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms.* Wiley InterScience, 2004.
39. L.I. Kuncheva. Clustering-and-selection model for classifier combination. In *Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 2000. Proceedings. Fourth International Conference on*, volume 1, pages 185–188 vol.1, 2000.
40. L.I. Kuncheva. A theoretical study on six classifier fusion strategies. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):281–286, 2002.
41. Mark Last, Horst Bunke, and Abraham Kandel. A feature-based serial approach to classifier combination. *Pattern Analysis and Applications*, 5(4):385–398, 2002.

42. D.-S. Lee. *Theory of Classifier Combination: The Neural Network Approach.* Ph.D Thesis, SUNY at Buffalo, 1995.
43. E. Mandler and J. Schuermann. Combining the Classification Results of Independent Classifiers Based on the Dempster/Shafer Theory of Evidence. In L.N. Kanal E.S. Gelsema, editor, *Pattern Recognition and Artificial Intelligence*, pages 381–393, 1988.
44. A.E. Rosenberg and S. Parthasarathy. Speaker background models for connected digit password speaker verification. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 81–84 vol. 1, 1996.
45. R. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.
46. R. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.
47. Glenn Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, 1976.
48. C. E. Shannon. A Mathematical Theory of Communication. *Bell System Tech. J.*, 27(623-656):379–423, 1948.
49. K. Sirlantzis, S. Hoque, and M. C. Fairhurst. Trainable Multiple Classifier Schemes for Handwritten Character Recognition. In *3rd International Workshop on Multiple Classifier Systems (MCS)*, pages 169–178, Cagliari, Italy, 2002. Lecture Notes in Computer Science, Springer-Verlag.
50. R. Tibshirani. Bias, variance and prediction error for classification rules. Technical Report 41, Department of Statistics, University of Toronto, 1996.
51. S. Tulyakov and V. Govindaraju. Classifier combination types for biometric applications. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), Workshop on Biometrics*, New York, USA, 2006.
52. S. Tulyakov and V. Govindaraju. Identification model for classifier combinations. In *Biometrics Consortium Conference*, Baltimore, MD, 2006.
53. K. Tumer and J. Ghosh. Linear and order statistics combiners for pattern classification. In A. J.C. Sharkey, editor, *Combining Artificial Neural Nets: Ensembles and Multi-Net Systems*, pages 127–162. Springer-Verlag, London, 1999.
54. V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.
55. V.N. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.
56. O. Velek, S. Jaeger, and M. Nakagawa. A New Warping Technique for Normalizing Likelihood of Multiple Classifiers and its Effectiveness in Combined On-Line/Off-Line Japanese Character Recognition. In *8th International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pages 177–182, Niagara-on-the-Lake, Canada, 2002.
57. O. Velek, S. Jaeger, and M. Nakagawa. Accumulated-Recognition-Rate Normalization for Combining Multiple On/Off-line Japanese Character Classifiers Tested on a Large Database. In *4th International Workshop on Multiple Classifier Systems (MCS)*, pages 196–205, Guildford, UK, 2003. Lecture Notes in Computer Science, Springer-Verlag.
58. W. Wang, A. Brakensiek, and G. Rigoll. Combination of Multiple Classifiers for Handwritten Word Recognition. In *Proc. of the 8th International Workshop on*

Frontiers in Handwriting Recognition (IWFHR-8), pages 117–122, Niagara-on-the-Lake, Canada, 2002.

59. D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–260, 1992.
60. K. Woods, Jr. Kegelmeyer, W.P., and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(4):405–410, 1997.
61. L. Xu, A. Krzyzak, and C. Y. Suen. Methods for combining multiple classifiers and their applications to handwriting recognition. *IEEE transactions on System, Man, and Cybernetics*, 23(3):418–435, 1992.