Classifier Combination Types for Biometric Applications

Sergey Tulyakov and Venu Govindaraju
Center for Unified Biometrics and Sensors (CUBS)
SUNY at Buffalo, USA
tulyakov@cubs.buffalo.edu

Abstract

In this paper we present four types of classifier combinations determined by the numbers of trained combining functions and their input parameters. We discuss the usage of these combination types in biometric applications and give an example of suboptimal combination as a result of choosing non-appropriate combination type. Finally, we present results of combinations in biometric identification systems utilizing similar methods, but related to different combination types.

1. Introduction

With the growth of classifier combination field it became important to categorize different approaches to classifier combinations. First of all we might distinguish combinations dealing with ensemble classifiers and non-ensemble classifiers. Usually ensemble classifiers are generated using subsets of training data or subsets of features, and we assume that there is a large number of classifiers to be combined. Only limited information about each classifier in the ensemble can be learned, and ensemble combinations are mostly limited to fixed combination rules. Non-ensemble classifiers represent traditional individually trained or constructed classifiers, and we assume that there is a small number of such classifiers in the combination. The behavior of each classifiers can be learned from the training data, and combination methods involve some training. Biometric applications usually require combinations of non-ensemble classifiers or matchers.

Another distinction between classifier combination methods is whether they operate on the confidence scores of classifiers assigned to different classes or on some internal classifiers' features. Generally, such features might not be available to system integrators who might simply treat biometric matchers as black boxes. Also, such features can be of limited use as, for example, fingerprint minutia positions. Thus we assume in this paper that we are working with confidence scores output by classifiers.

The third distinction between classifier combinations is based on the type of classifiers' outputs[22]:

- Type I : output only a single class. This type can also include classifiers outputting a subset of classes to which the input pattern can belong. This is equivalent to the classifier assigning a confidence of 0 or 1 to each class.
- Type II: output a ranking for all classes. Each class is given a score equal to its rank - $1, 2, \ldots, N$.
- Type III: output a score for each class, which serves as a confidence measure for the class to be the true class of the input pattern. Scores are arbitrary real numbers.

If the combination involves different types of classifiers, their output is usually converted to any one of the above: to type I[22], to type II[9] or to type III[16]. In this paper we will assume that the classifier output is of type III.

In the next section we will introduce another categorization of classifier combinations which rather deals with the structural properties of combinations. As an example, consider the general scheme for classifier combination shown in figure 1. The final score for each class is derived from the scores received from all the classifiers for that class. This approach has low complexity, and many well known combination methods (e.g. weighted sum of scores) fall into this category. But, it is also possible to consider a more general form of combination where derivation of a final score for a particular class includes all classifier scores, for that class as well as for other classes [16]. A class confusion matrix can be used for the construction of such combination methods [22].

If classifiers deal with the small number of classes, then the dependencies between scores assigned to different classes can be learned and used for combination purposes. Xu et al. [22] used class confusion matrices for deriving belief values and integrated these values into combination algorithms in the digit classification problem. Learning class dependencies requires significant number of training samples of each class. Thus, if considered application has small

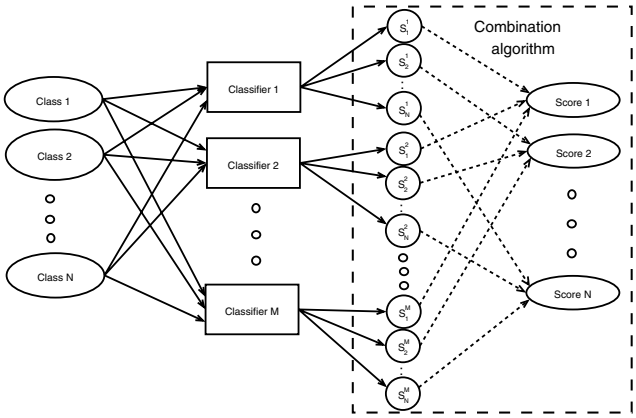


Figure 1. Classifier combination takes a set of s_i^j - score for class i by classifier j and produces combination scores S_i for each class i.

number of classes and sufficient number of training samples for each class, such combinations can be superior to the low complexity combinations of figure 1.

The number of classes, i.e. the number of enrolled persons, in biometric applications is usually large, and multiple training samples of each class are difficult to obtain. In addition, the database of enrolled persons can be frequently changed, and this makes learning class relationships infeasible. Consequently, combination approaches in biometric systems consider only matching scores related to the single person in order to derive a combined score for that person. The question is whether we can improve a combination performance if we somehow account for class dependencies in the classifiers' scores.

In the next section we identify four types of combinations depending on the number of matching scores they consider and on the number of trainable combination functions. One of these types (medium II complexity type) seems to be the most appropriate combination type for biometric matchers in identification systems. Then we give a review of existing approaches to classifier combinations in the context of presented framework. Next we give an example of improper choice of the combination type resulting in the decrease of system performance. Finally, we present a results of combination experiments in biometric identification system using different combination types.

2. Complexity Types of Classifier Combinations

Combination algorithms (combinators) can be separated into 4 different types depending on the number of classifier's scores they take into account and the number of combination functions required to be trained. As in Figure 1 i is the index for the N classes and j is the index for the M classifiers.

1. Low complexity combinators: $S_i = f(\{s_i^j\}_{j=1,\ldots,M})$. Combinations of this type require only one combination function to be trained, and the combination function takes as input scores for one particular class as parameters.

2. Medium complexity I combinators: $S_i = f_i(\{s_i^j\}_{j=1,\ldots,M})$. 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.

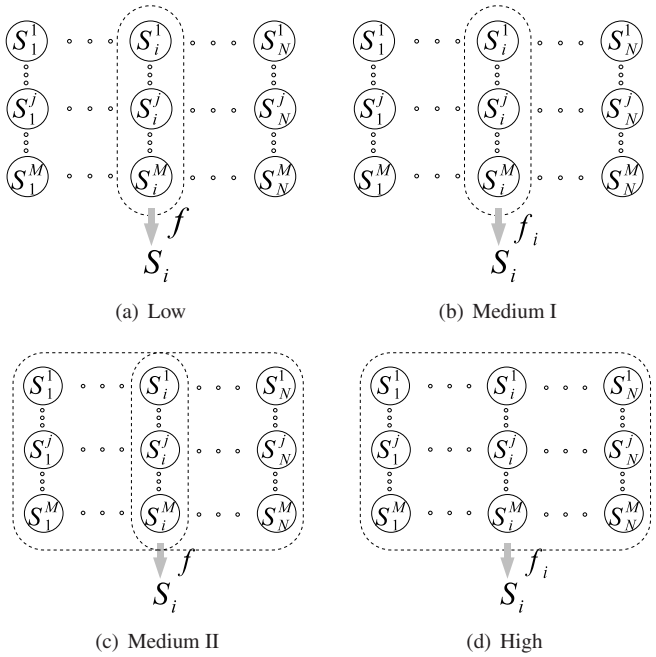
3. Medium complexity II combinators: $S_i = f(\{s_i^j\}_{j=1,\ldots,M}, \{s_k^j\}_{j=1,\ldots,M;k=1,\ldots,N,k\neq i})$. This function takes as parameters not only the scores related to this class, but all output scores of classifiers. Combination scores for each class are calculated using the same function, 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 combinators: $S_i = f_i(\{s_k^j\}_{j=1,\ldots,M;k=1,\ldots,N})$. Functions calculating final scores are different for all classes, and they take as parameters all output base classifier scores.

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. From the perspective of this framework, the main effort in solving classifier combination problem consists in a justification for a particular chosen complexity type of combination and providing any special modifications to generic classifiers compensating for this chosen complexity type. The choice of used generic classifier or combination function is less important than the choice of the complexity type.

In order to illustrate the different combination types we can use a matrix representation. 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. The illustration of each combination type functions is given in Figure 2. In order to produce the combined score S_i for



(a) Low (b) Medium I

(c) Medium II (d) High

Figure 2. The range of scores considered by each combination type and combination functions.

class i low complexity combinations (a) and medium I complexity (b) combinations consider only classifier scores assigned to class i (column i). Medium II (c) and high complexity (d) combinations consider all scores output by classifiers for calculating a combined score S_i for class i.

Low (a) and medium II (c) complexity combinations have the same combination functions f irrespective of the class for which the score is calculated. Note that medium II complexity type combinations have scores related to a particular class in a special consideration as indicated by the second ellipse around these scores. We can think of these combinations as taking two sets of parameters - scores for a particular class, and all other scores. The important property is that combination function f is same for all classes, but the combined scores S_i differ, since we effectively permute function inputs for different classes. Medium I (b) and high (d) complexity combinations have combining functions f_i trained differently for different classes.

Figure 3 illustrates the relationships between presented complexity types of combinations. Medium complexity types are subsets of high complexity combinations, and the set of low complexity combinations is exactly the intersection of sets of medium I and medium II combination types. In order to avoid a confusion in terminology we will henceforth assume that a combination method belongs to a particular type only if it belongs to this type and does not belong to the more specific type.

It is interesting to compare our combinations types with previous categorization of combination methods by

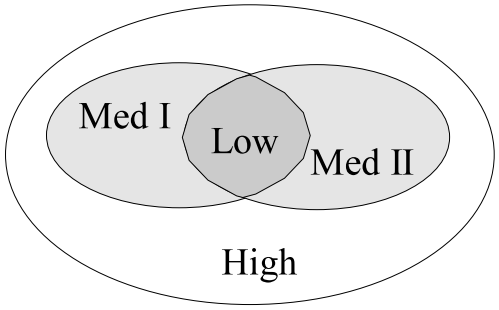


Figure 3. The relationship diagram of different combination complexity types.

Kuncheva et al.[15]. In that work the score matrix has names 'decision profile' and 'intermediate feature space'. It seems that using term 'score space' makes more sense here. Kuncheva's work also separates combinations into 'class-conscious' set which corresponds to the union of 'low' and 'medium I' complexity types, and 'class-indifferent' set which corresponds to the union of 'medium II' and 'high' complexity types. Again these terms might not be suitable since we can think of a combination method as being 'class-conscious' if each class has its own combination function ('medium I' and 'high' complexity types), and 'class-indifferent' if combination functions are same for all classes ('low' and 'medium II' complexity types). The continuation of this work [14] gave an example of the weighted sum rule having three different numbers of trainable parameters (and accepting different numbers of input scores), which correspond to 'low', 'medium I' and 'high' complexity types.

In contrast to Kuncheva's work, our categorization of combination methods is more general since we are not limiting ourselves to simple combination rules like weighted sum rule. Also we consider an additional category of 'medium II' type, which is missed there. An example of 'medium II' combinations are two step combination algorithms where in the first step the scores produced by a particular classifier are normalized with the participation of all scores of this classifier, and in the second step scores are combined by a function from 'low' complexity type. Thus all scores participate in the combined score for each class, and if normalization function is same for all classes, then the combination function is also the same.

3. Previous Research

Biometric applications are traditionally separated into verification and identification systems. Verification systems usually have only the confidence scores related to one person available for combination, and their task is to separate genuine and impostor verification attempts. Thus, for

verification systems, score matrices in Figure 2 have only one column, and all types collapse into one. Identification systems, on the other hand, can have matching confidence scores produced for all enrolled persons, and score matrices with $N > 2$ columns. The choice of combination type becomes important for identification systems. Note that most traditional pattern classification problems, such as optical character recognition, also have the number of classes bigger than 2, so the presented framework is directly applicable to those problems.

3.1. Low and Medium I Type Combinations

The most frequent approach to combinations in identification systems is the use of some combination function f to produce combined score for each class from classifiers' scores assigned to the same class. In our combination framework such combinations are of the low complexity type. Combination functions can also be user specific - f_i [12, 6] (medium I complexity type). These are in fact the same combination methods which are used in biometric verification problems where only scores related to one person are available for combination.

For example, suppose we have a score combination algorithm used for verification task. It can be represented as a function $f(\{s^j\}_{j=1,\ldots,M})$ combining scores s^j assigned by M matchers to a test sample. The test sample is verified if the value of the function is bigger than some predefined threshold. The direct extension of this combination algorithm to the identification systems will compute the confidences of matching test sample to any enrolled person i, $f(\{s_i^j\}_{j=1,\ldots,M})$, and finding a maximum among these combined scores. Note, that combination function is same for all persons i, and it takes as input parameters only scores related to this person. Thus, such extension of verification task combination algorithm to identification systems is indeed of low complexity type.

We can also note, that the training of such function f will usually involve separate genuine and impostor score sets $\{s^1_{gen_i}, \ldots, s^M_{gen_i}\}$ and $\{s^1_{imp_i}, \ldots, s^M_{imp_i}\}$, and the dependencies between genuine and impostor scores from the same matcher $s^j_{gen_i} \sim s^j_{imp_i}$ will be discarded. Such score dependence is also discarded when combination algorithm utilizes some performance characteristics of biometric matchers which treat genuine and impostor scores separately: score densities, FAR and FRR curves, and derived from them ROC curve.

The main reason for using medium II and high complexity combinations is to account for the dependencies between matching scores assigned to different classes by the same matcher. As our previous experiments show[21], such dependence does exist in many identification applications. In the remaining part of this section we review existing approaches to utilize score dependence - rank based classifier combinations and combinations involving different score normalization techniques.

3.2. Rank based approaches

The frequent approach to combination in identification systems is to use rank information of the scores. This approach transforms combination problems with measurement level output classifiers to combination problems with ranking level output classifiers ([22]). T.K. Ho has described classifier combinations on the ranks of the scores instead of scores themselves by arguing that ranks provide more reliable information about class being genuine [8, 9]. Thus, if the input image has low quality, then the genuine score, as well as the impostor scores will be low. Combining low score for genuine class with other scores could confuse a combination algorithm, but the rank of the genuine class remains to be a good statistic, and combining this rank with other ranks of this genuine class should result in true classification. Brunelli and Falavigna [4] considered a hybrid approach where traditional combination of matching scores is fused with rank information in order to achieve identification decision. Hong and Jain [10] consider ranks, not for combination, but for modeling or normalizing classifier output score. Saranli and Demirekler [20] provide additional references for rank based combination and a theoretical approach to such combinations.

Rank-based methods are examples of the medium II complexity type combinations. Recall that combinations of these type consider possibly all output classifiers' scores, and use the same combination function irrespective of the class. Indeed, rank based methods take into account all scores output by each classifier in order to calculate ranks. The ranks are combined at the second stage using some non-class specific combination functions (e.g. Borda count). Thus combination functions are indeed independent of the class, and there is only one combination function applied to all classes.

Despite the apparent simplicity of rank based combination methods, they are placed in the higher complexity type than previously mentioned low complexity combinations. As many authors suggest, these methods do provide a better performance in identification systems. The problem with rank based methods, however, is that the score information is somewhat lost. It would be desirable to have a combination method which retains the score information as well as the rank information.

3.3. Score normalization approaches

Ranking of the matching scores is somewhat similar to the score normalization. Usually score normalization [11] means transformation of scores based on the classifier's score model learned during training, and each score is transformed individually using such a model. Thus the other

scores output by a classifier during the same identification trial are not taken into consideration. If these normalized scores are later combined by low complexity combination, then the resulting total combination algorithm will still be of low complexity. On the other hand, rank based normalization considers all scores of a classifier in order to derive a normalized score for a particular class, and thus results in higher complexity combinations.

Some score normalization techniques can use a whole set of scores output by classifier. For example, Kittler et al. [13] normalize each score by the sum of all other scores before combination. The combinations employing such normalizations can be considered as medium II complexity type combinations. However, Altincay and Demirekler [2] note that useful classification information gets lost during such normalizations.

Score normalization techniques have been well developed in the speaker identification problem. Cohort normalizing method [19, 5] considers a subset of enrolled persons close to the current test person in order to normalize the score for that person by a log-likelihood ratio of genuine (current person) and impostor (cohort) score density models. [3] separated cohort normalization methods into cohorts found during testing (constrained) and cohorts dynamically formed during testing (unconstrained cohorts). Normalization by constrained cohorts followed by low level combination amounts to medium I combination types, since whole combination method becomes class-specific, but only one matching score of each classifier is utilized. On the other hand, normalization by unconstrained cohorts followed by low level combination amounts to medium II or high complexity combinations, since now potentially all scores of classifiers are used, and combination function can be class-specific or non-specific.

The related normalization techniques are Z(zero)- and T(test)- normalizations [3, 17]. Z- normalization is similar to a constrained cohort normalization, since it uses impostor matching scores to produce a class specific normalization. Thus Z-normalization used together with low complexity combinator results in medium I combination. T-normalization uses a set scores produced during single identification trial, and used together with low complexity combinator results in medium II combination (note that this normalization is not class-specific).

Medium II combinations seem to be the most appropriate type of combinations for biometric applications. Indeed, it is usually hard to train class-specific combination types of medium I and high complexity since we might have only single template per user available. As an example justifying medium II combinations in biometrics, [7] argued for applying T-normalizations in face verification competition. Another example of normalization leading to medium II combinations is identification model[21] developed for thresholding results in identification system. Whereas T-normalization is a non-trainable algorithm, identification model is trained for each classifier in order to account for the dependencies between genuine and impostor scores. We will use similar identification model in our experiments on combining biometric matchers in identification systems below.

4. Low Complexity Combinations in Identification System

This section contains an example of combining two classifiers with score dependencies in the identification system. We consider combinations of low complexity type and of medium II complexity type. The goal of this section is to show that low complexity combinations fail to account for score dependencies and produce suboptimal combination. In fact, the combined classifier performs worse than one classifier used for combination.

Consider a combination of two hypothetical matchers in two class identification system both of which have the distributions of genuine and impostor scores as shown on figure 4. Suppose that the first matcher produces both scores independently drawn from these distributions (one score is genuine, the other is impostor). For the second matcher, suppose that it produces dependent scores in each identification attempt where it is always $s_{imp} = s_{gen} - 1$. Note that both matchers have same distributions of genuine and impostor scores, but second matcher is optimal for identification system, since top score is always genuine.

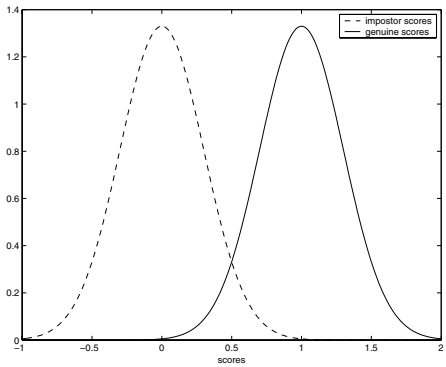


Figure 4. Hypothetical densities of matching(genuine) and non-matching(impostors) scores.

Assume that these matchers are independent. Let the upper score index refer to the matcher producing this score; s_i^j is the score for class i assigned by the classifier j. From our construction we know that the second matcher always outputs genuine score on the top. So the optimal combination algorithm simply look at scores s_1^2 and s_2^2 output by the second matcher and classifies the input as $\arg\max_i s_i^2$. Such

a combination considers the whole set of scores produced by the second matcher, and thus belongs to the medium II complexity type.

Now suppose we can only use combinations of the low complexity type. These combinations use some function f to combine scores assigned to the same class and classify the input as a class producing the best combined score: $\arg\max_i f(s_i^1, s_i^2)$. The training of the combination function f can be performed only by taking sample pairs of scores (s_i^1, s_i^2), with some pairs belonging to the genuine matching scores and other pairs belonging to impostor matching scores. Even though we might have our scores originating from identification trials $\{(s_1^1, s_1^2), (s_2^1, s_2^2)\}$, we still have to separate them into genuine and impostor score pairs and use them separately for training. The information about the dependence of scores output by any classifier during one identification trial is simply discarded.

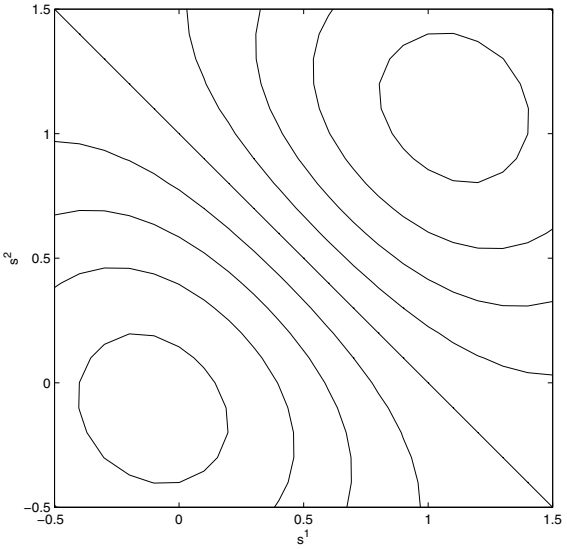


Figure 5. Optimal decision surfaces for low complexity combinations.

Pairs of scores (s_i^1, s_i^2) belonging to genuine and impostor matchings could be displayed in the $s^1 \times s^2$ score space. In our example, impostor scores are distributed as a Gaussian centered at $(0, 0)$, and genuine scores are distributed as a Gaussian centered at $(1, 1)$. Figure 5 shows the decision contours for optimal classification of genuine and impostor matches. The optimal classification decision in this space looks at the ratios of genuine and impostor densities at points (s_1^1, s_1^2) and (s_2^1, s_2^2) and classify the sample as the class giving the bigger ratio (the proof of this is similar to the derivation of likelihood ratio rule we give in the next section). The contours in Figure 5 are exactly the curves where such ratio is constant.

Now, suppose we conduct a testing of this combination method, and the test sample is $(s_1^1, s_1^2) = (-0.1, 1.0)$, $(s_2^1, s_2^2) = (1.1, 0)$. We know from our construction that class 1 is the genuine class, since the second matcher assigned score 1.0 to it and 0 to the second class. But its score pair $(1.1, 0)$ is located just above the diagonal $s^1 + s^2 = 1$, and the score pair $(-0.1, 1.0)$ corresponding to class 1 is located just below this diagonal. Hence class 2 has bigger ratio of genuine to impostor densities than class 1, and the optimal low complexity method would incorrectly classify class 2 as the genuine class.

We can also show that this sample will be incorrectly classified by the following reasoning. Combination function f should be symmetrical in its arguments since distributions of genuine and impostor scores s^1 and s^2 are identical. We also know that the genuine scores are generally higher than impostor scores, thus function f should be increasing in its arguments (higher score should result in higher combined score output by f). So, for the first class $f(-0.1, 1.0) = f(1.0, -0.1)$, which should be smaller than the value for second class $f(1.1, 0)$.

We have presented an example of the identification system with two matchers, which has optimal performance by utilizing combinations from the medium II complexity type, and suboptimal performance if combinations from low complexity type are used. If at the beginning we considered an identification system with only the second matcher (having the optimal performance) and added another matcher (suboptimal), and used only combinations of the low complexity type, we would have decreased the performance of this identification system.

This somewhat contradicts the generally accepted rule that incorporating additional classifiers into the recognition system should not decrease system performance (at least theoretically). If combination decreases system performance, it is usually explained by the small training set and training errors or by incorrectly chosen combination function. It does not matter what low complexity combination function is chosen in our example, the performance will still be worse than before combination. As our example shows, such decrease in performance can be caused by the improper choice of the combination complexity type.

Combination algorithms of low complexity type simply discard the dependency information between scores assigned to all classes by one classifier. The example illustrates, that if such information is discarded and low complexity type combinations are used instead of medium II complexity type combinations, then the combination can result in a worse performance than the performance of the single involved classifier. Interestingly, Rao [18] proved that the fusion performance can not be worse than the performance of any involved classifier, if the system possesses the so called isolation property, that is, single classifiers are included in a set of possible combinations. In our ex-

ample low complexity combinations possess the isolation property, but the performance of the combination is worse than the performance of a single classifier. However, our example does not contradict Rao's work. Rao considered two class classifiers outputting a single score differentiating two classes, and for such combinations all the complexity types degenerate into one low complexity type. In our case, we assume that classifiers output at least two scores each, and we truly have these 4 different combination types. The performance decrease comes from the inability of low complexity combinations to properly model the score relationships.

5. Experiments

In order to test presented theory we investigated the combination of fingerprint and face matchers using NIST BSSR1 biometric score database [1]. We used two subsets of fingerprint scores: li (left index) and ri (right index), and two subsets of face scores from two face matchers C and G. Since we wanted to consider the case of independent matchers we performed four sets of experiments on combining fingerprint and face scores : li combined with C, li combined with G, ri combined with C, and ri combined with G.

Results are presented in Table 1. The columns represent the combination method. 'Low' is the method of reconstructing densities of genuine and impostor score pairs, and performing Bayesian classification using this densities. This approach discards score dependencies, and it is of low complexity type. 'Medium II' is also Bayesian classification method, but the variant of identification model[21] is used to model score dependencies and thus to normalize scores. Specifically, this model employs the statistics of the score set produced during one identification trial t_s = 'best score besides s', and reconstructs densities of pairs (s, t_s) for s genuine and s impostor. The ratio of such densities serves as normalized score. All the densities are reconstructed using original scores linearly normalized to interval $[0, 1]$, and the kernel sizes are derived using the maximum likelihood method.

Matchers	Number of tests	Low	Medium II
li & C	516	5	4
li & G	517	9	6
ri & C	516	3	2
ri & G	517	3	2

Table 1. Experiments on combinations in identification systems. Entries are the numbers of failed test identification trials.

All experiments were performed in leave-one-out framework. The numbers in the tables are the numbers of failed tests, and total number of tests is also given. Failed test means that the impostor got the best combination score in this particular identification attempt.

The reason for improvement appears to be the ability of used identification model to account for score dependencies in identification trials. The considered biometric scores indeed showed some dependency: the correlation between genuine and best impostor scores in identification trials was around $0.10 - 0.15$ for these score sets. As the example in the previous section illustrates, low complexity combinations discard this dependency information and produce suboptimal results.

6. Conclusion

We can view a classifier combination problem as a secondary classification problem in the score space. If the number of classes or the number of classifiers is too large for adequate training of the classification algorithm, we have to consider constrained combination algorithms of low, medium I or medium II type. These types of combinations arise naturally from the assumed meaning of scores: 'produced by jth classifier and related to ith class'. Thus, the problem of classifier combination can be defined as a problem of choosing appropriate reduction to a lower complexity class, and possibly accounting for the discarded dependencies between scores.

As we showed by the constructed example, by reducing complexity of combination from medium II to low we can loose important information about dependence between scores produced during single identification trial. The experiments on real biometric scores show that such dependence exists, and taking it into consideration can improve the performance of the combination algorithm.

References

[1] Nist biometric scores set. http://www.nist.gov/biometricscores/. 7

[2] H. Altincay and M. Demirekler. Undesirable effects of output normalization in multiple classifier systems. *Pattern Recognition Letters*, 24(9-10):1163–1170, 2003. 5

[3] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas. Score normalization for text-independent speaker verification systems. *Digital Signal Processing*, 10(1-3):42–54, 2000. 5

[4] R. Brunelli and D. Falavigna. Person identification using multiple cues. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(10):955–966, 1995. 4

[5] J. Colombi, J. Reider, and J. 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. 5

[6] J. Fierrez-Aguilar, D. Garcia-Romero, J. Ortega-Garcia, and J. Gonzalez-Rodriguez. Bayesian adaptation for user-dependent multimodal biometric authentication. *Pattern Recognition*, 38(8):1317–1319, 2005. 4

[7] P. Grother. Face recognition vendor test 2002 supplemental report, nistir 7083. Technical report, 2004. 5

[8] T. K. Ho. *A Theory of Multiple Classifier Systems And Its Application to Visual Word Recognition*. Ph.d thesis, SUNY Buffalo, 1992. 4

[9] T. K. Ho, J. Hull, and S. Srihari. Decision combination in multiple classifier systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(1):66–75, 1994. 1, 4

[10] L. Hong and A. Jain. Integrating faces and fingerprints for personal identification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(12):1295–1307, 1998. 4

[11] A. Jain, K. Nandakumar, and A. Ross. Score normalization in multimodal biometric systems. *Pattern Recognition*, 38(12):2270–2285, 2005. 4

[12] A. Jain and A. Ross. Learning user-specific parameters in a multibiometric system. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–57–I–60 vol.1, 2002. 4

[13] J. Kittler, M. Hatef, R. Duin, and J. Matas. On combining classifiers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(3):226–239, 1998. 5

[14] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley InterScience, 2004. 3

[15] L. I. Kuncheva, J. C. Bezdek, and R. P. W. Duin. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34(2):299–314, 2001. 3

[16] D.-S. Lee. *Theory of Classifier Combination: The Neural Network Approach*. Ph.D Thesis, SUNY at Buffalo, 1995. 1

[17] J. Mariethoz and S. Bengio. A unified framework for score normalization techniques applied to text independent speaker verification. *IEEE Signal Processing Letters*, 12, 2005. 5

[18] N. Rao. On fusers that perform better than best sensor. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(8):904–909, 2001. 6

[19] A. 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. 5

[20] A. Saranli and M. Demirekler. A statistical unified framework for rank-based multiple classifier decision combination. *Pattern Recognition*, 34(4):865–884, 2001. 4

[21] S. Tulyakov and V. Govindaraju. Combining matching scores in identification model. In *8th International Conference on Document Analysis and Recognition (ICDAR 2005)*, Seoul, Korea, 2005. 4, 5, 7

[22] 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. 1, 4