A COMPLEXITY FRAMEWORK FOR COMBINATION OF CLASSIFIERS IN VERIFICATION AND IDENTIFICATION SYSTEMS

By

Sergey Tulyakov

May 2006

A DISSERTATION SUBMITTED TO THE

FACULTY OF THE GRADUATE SCHOOL OF STATE

UNIVERSITY OF NEW YORK AT BUFFALO

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Department of Computer Science and Engineering

Acknowledgments

I would like to thank my advisor Dr. Venu Govindaraju for directing and supporting this dissertation. His constant encouragement made the research going, and his numerous suggestions and corrections were very helpful in preparing this thesis. I am grateful to Dr. Sargur Srihari and Dr. Peter Scott for taking the time to serve on my dissertation committee. I would also like to thank Dr. Arun Ross for detailed reading of the draft, and advising multiple ways for its improvement.

During the years of studying and working on the dissertation I have enjoyed the interactions with many people of the Center of Excellence in Document Analysis and Recognition (CEDAR) and the Center for Unified Biometrics and Sensors (CUBS). In particular, I would like to thank Dr. Petr Slavik for introducing me to this interesting field of research and supervising my earlier work. I had many fruitful discussions with Dr. Krassimir Ianakiev and Dr. Jaehwa Park on word and character recognition, and with Dr. Tsai-Yang Jea on fingerprint matching. I enjoyed being in a team with Dave Bartnik, Phil Kilinskas and Dr. Xia Liu on a fingerprint identification system. I truly appreciate the support and encouragement of many people I worked with at CEDAR and CUBS.

I would also like to thank my previous advisors in mathematics, Dr. Aleksandr I.

Shtern and Dr. E. Bruce Pitman. The mathematical background turned out to be very useful asset during the work on dissertation.

Finally, I extend my thanks to my parents, Viktor and Natalia, who always believed in me and the success of this work. Also I express the gratitude to all my friends, and especially to Liana Goncharuk, Vitaliy Pavlyuk and Jaroslaw Myszewski.

Abstract

In this thesis we have developed a classifier combination framework based on the Bayesian decision theory. We study the factors that lead to successful classifier combination and identify the scenarios that can benefit from specific combination strategies. The classifier combination problem is viewed as a second-level classification task where the scores produced by classifiers can be taken as features. Thus any generic pattern classification algorithm (neural networks, decision trees and support vector machines) can be used for combination. However, for certain classifier combination problems such algorithms are not applicable, or lead to performance which is worse than specialized combination algorithms. By identifying such problems we provide an insight into the general theory of classifier combination. We introduce a complexity categorization of classifier combinations, which is used to characterize existing combination approaches, as well as to propose new combination algorithms.

We have applied our proposed theory to identification systems with large number of classes as is often the case in biometric applications. Existing approaches to combination for such systems use only the matching scores of one class to derive a combined score for that class. We show both theoretically and experimentally that these approaches are inferior to methods which consider the output scores corresponding to

all the classes. We introduce the identification model which accounts for the relationships between scores output by one classifier during a single identification trial. This allows the construction of combination methods which consider a whole set of scores output by classifiers in order to derive a combined score for any one class. We also explore the benefits of utilizing the knowledge of classifier independence in combination methods.

Contents

Chapter 1

Introduction

The ability to combine decisions from different sources is important for any recognition system. For example, a deer in the forest produces both visual and auditory signals. Consequently, the predator's brain processes visual and auditory perceptions through different subsystems and combines their output to identify the prey. Humans perform subconscious combination of sensory data and make decisions regularly. A friend walking at a distance can be identified by the clothing, body shape and gait. It was noted in psychology literature that significant degree of human communication takes place in a non-verbal manner[5]. Thus in a conversation we make sense not only from the spoken words, but also from gestures, face expressions, speech tone, and other sources.

More deliberate decision making also involves combination of information from multiple sources. For example, military and economic decisions are committed only after considering different sources of information about enemies or competitors. Court decisions are made after considering evidence from all sources and weighing their

individual strengths.

The field of pattern recognition is about automating such recognition and decision making tasks. Thus the need to develop combination algorithms is fundamental to pattern recognition. It is generally agreed that using a set of classifiers and combining them somehow can be superior to the use of a single classifier[27]. The decisions of the individual experts are often conflicting [13], [28], [30], [37], [36], [44], and combining them is a challenging task.

1.1 Problem

The field of classifier combination research has expanded significantly in the last decade. It is now possible to divide the general problem into subareas based on the type of the considered combinations. Such a categorization will also allow us to define precisely the main area of our research.

1.1.1 Combinations of Fixed Classifiers and Ensembles of Classifiers

The main division is based on whether 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.We will focus primarily on the first type of combinations.

1.1.2 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, any negative effect is usually compensated by the lower complexity of the combination method and superior training of the final system. We will only consider classifier combinations at the decision level.

1.1.3 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[63]:

- 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[63], to type II[28] or to type III[46]. In this thesis we will assume that the classifier output is of type III.

Among combinations of classifiers of this type we could find combinations with fixed structure (different voting schemes, Borda count, sum of scores [63, 37]) or combinations that can be trained using available training samples (weighted vote, logistic regression[28], Dempster-Shafer rules [63] and neural network[46]).

1.1.4 Considered Applications

The efforts to automate the combination of expert opinions have been studied extensively in the second half of the twentieth century[16]. 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[14]. 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, [14] considers combinations of expert's estimations of probability density functions.

The pattern classification field developed around the end of twentieth century deals with more specific problems of assigning input signal to two or more classes. In the current thesis we consider combinations of classifiers from a pattern classification perspective. The combined experts are classifiers and the result of the combination is also a classifier. The output types of classifiers were described in the previous section. The unifying feature of these outputs is that they could be all 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, where the function is optimal in some sense, e.g. minimizing the misclassification cost. The pattern classification uses mostly statistical methods, and classifier combination field employs statistics as well.

The applications of pattern classification include image classification, e.g OCR (optical character recognition) and word recognition, speech recognition, person authentication by voice, face image, fingerprints and other biometric characteristics. In our thesis we will mostly consider the biometric person authentication applications. Though our proposed classifier combination framework (chapter 2) is applicable to all applications in general, parts of the thesis might require assumptions specific to biometric authentication. For example, chapter 3 uses the assumption of classifier independence which is true for biometric matchers of different modalities, and chapters 4 and 5 assume large number of classes which is appropriate in person identification systems.

1.2 Objective

The main motivation for our research is to develop an efficient algorithm for combining biometric matchers in person identification systems. After studying multiple combination methods for biometric data [37, 7, 32, 34], we have observed that the considered combination functions use only the matching scores of a particular person to obtain a final combined score for that person. This is different from combinations considered in the OCR field where a final matching score for a particular character

is derived from the matching scores of all the characters [46]. Thus we wanted to investigate whether similar combinations could be effective in biometric identification systems. As a result we developed a categorization of classifier combinations based on the number of considered scores and on the variability of the combination function. The main objective of this thesis is to develop a framework for classifier combination based on this categorization.

The framework should help a practitioner to choose a proper combination method for a particular classifier combination problem. The traditional approach for choosing a combination method is to try a few methods and find the one which shows the best performance on a test set. In our work we presume that all combinations methods are roughly equivalent if they possess a property of universal approximation. Thus, instead of studying whether a particular combination method (say, SVM or neural network) has better approximating abilities, we are more interested in the type of this method, that is how many input parameters it considers and whether its training is different for each class. The purpose of this thesis is to investigate whether the choice of the combination type is more important than the choice of the used universal approximator.

We assume that the combination method is in practice a classifier acting on the outputs of combined classifiers. The question is whether this combining classifier should be any different from traditional classifiers used in the pattern classification field. Pattern classifiers working in a feature vector space do not make any distinction among features. If we find that there are some distinctions or connections between the outputs of the combined classifiers (on which combination classifier operates), the combinator could utilize such connections in order to have a more efficient combination method. One such connection exists when we consider biometric matchers of different modalities as the output scores related to different matchers are statistically

independent. We will investigate in this thesis whether utilizing such knowledge can improve combination results.

1.3 Outline of the Dissertation

In the second chapter we will introduce the framework for classifier combination. The classifier combination task is approached from the complexity point of view. We describe the challenges, categorize them and develop possible solutions. This chapter also discusses issues with classifier ensembles.

In the third chapter we consider an example of combinations for which we have the knowledge of the independence of classifiers' scores. Combinations of this kind can be used in biometric person authentication tasks. The chapter deals with the question of how this additional knowledge can improve the combination algorithm.

The fourth chapter discusses a problem of combining recognition scores for different classes produced by one recognizer during a single recognition attempt. Such scenarios occur in identification problems (1:N classification problems) with large or variable N. By using artificial examples we show that an intuitive solution of making the identification decision based solely on the best matching score is often suboptimal. We draw parallels with score normalization techniques used in speaker identification. We present three real life applications to illustrate the benefits of proper combination of recognition scores.

The fifth chapter approaches combination problems in cases where the number of classes is large as in biometric person identification, recognition of handwritten words and recognition of barcodes. We investigate the dependency of scores assigned to

different classes by any one classifier. We will utilize the independence model during classifier combinations and experimentally show the effectiveness of such combinations.

Finally, the sixth chapter contains the summary of our work and contributions made.

Chapter 2

Combination Framework

In order to produce a combined score for a particular class, combination algorithms usually use scores assigned by classifiers only to this particular class, although they could use the set of scores assigned to all classes [46]. For example, a neural network can be trained to operate only on scores related to a particular class, or on all the scores output by the classifiers. In this chapter we study both types of combinations. In fact, combination methods can be divided into 4 types based on the combination function complexity.

2.1 Score Combination Functions and Combination Decisions

Any approach to classifier combination essentially operates on the outputs of individual classifiers. For example, Dar-Shyang Lee [46] 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. One can also use a function or a rule to combine the classifier scores in a predetermined manner. It is generally accepted that using combination rules is preferable although there is no conceptual difference between two approaches.

Note that the final goal of classifier combination is to create a classifier which operates on the same type of input as base classifiers and separates the same types of classes. Using combination rules implies some final step of making 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 $S_i = f(\{s_i^j\}_{j=1,\ldots,M})$. 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.

Classifiers do not have to be necessarily constructed following the above described scheme, but in practice we see this theme present is 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 highest ratio is then assigned to a sample.

Combination rules f are usually some 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. Also, combination rules usually consider scores corresponding only to a particular class in order to calculate the final

score for the class, whereas generic classifiers can consider scores belonging to other classes as well. We will discuss this consideration of all scores in more detail in the next section.

In summary, simple combination rules f can be considered as a special type of classifier which operates on the base classifiers' scores. We will explore whether this approach has any advantage over the generic classifiers.

2.2 Complexity of Classifier Combinators

The general scheme for classifier combination is shown in figure 2.2.1. The final score for a 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 (Borda count, sum of scores) fall into this category. 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 [46]. A class confusion matrix can be used for the construction of such combination methods[63]. The disadvantage of this more general approach is that it requires enormous amount of training data.

2.2.1 Complexity of Combination Functions

We introduce the notion of complexity of combinators in this section. Intuitively, complexity should reflect the number of base classifier scores that influence the calculation of the combined scores, and how complicated combination functions really are. Previous approaches to define the complexity have included simply counting numbers

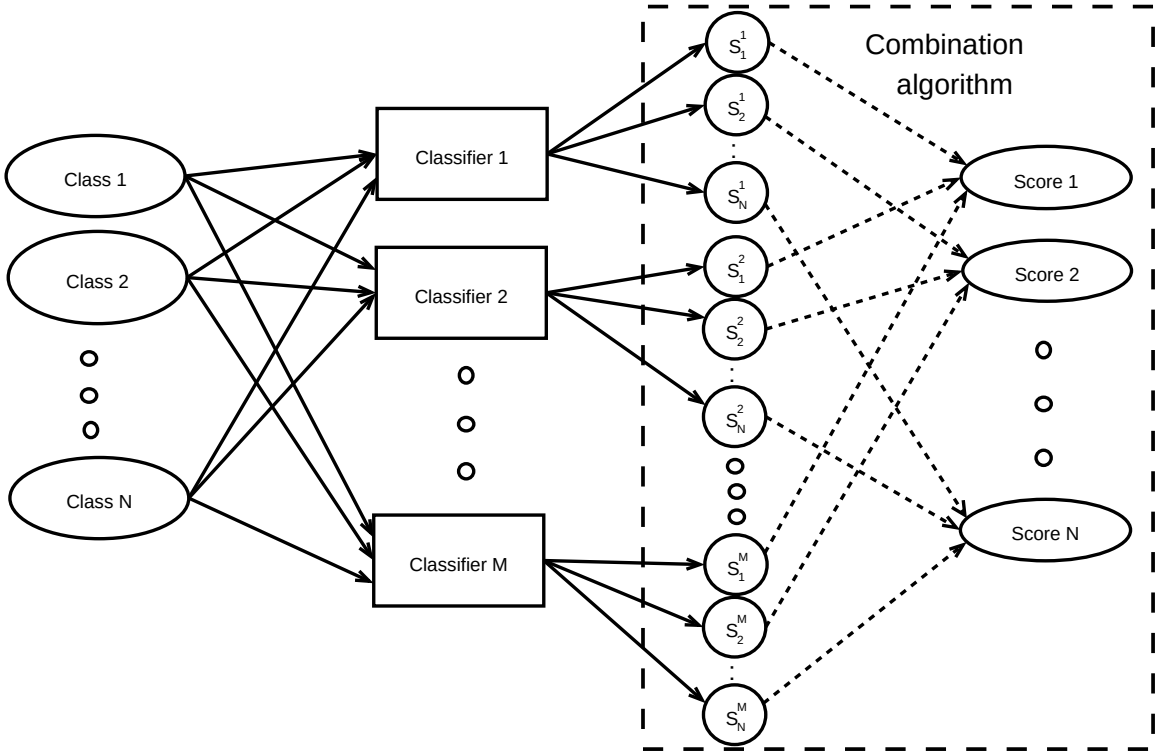


Figure 2.2.1: Classifier combination takes a set of s_i^j - score for class i by classifier j and produces combination scores S_i for each class i.

of trained function parameters, Kolmogorov's algorithmic function complexity and the VC (Vapnik-Chervonenkis) dimension [60].

Although the VC dimension seems to be the most appropriate complexity definition for our purpose, it does not take into account the number of input parameters being considered. Combination functions might not use all available base classifiers' scores for calculating final combination scores. For example, in order to calculate a final combined score for a particular class sum and product rule use only the base scores related to that particular class, and do not use scores related to other classes. We extend the concept of VC dimension to include the number of input parameters.

$C_f(K)$ can be taken as the VC dimension of a set of functions $\{f\}$ where the set contains functions non-trivially dependent on K input parameters.

Another drawback of the traditional definition of VC dimension is that it is defined for functions having output values in one-dimensional space. But the combination algorithm (Figure 2.2.1) generally would be a map $\{s_k^j\}_{j=1,...,M;k=1,...,N} \Rightarrow \{S_i\}_{i=1,...,N}$ into N-dimensional final score space, where N is the number of classes. In appendix A we extended the definition of VC dimension to functions having multidimensional outputs. The essence of the definition and related theorem is that the VC dimension H of an N dimensional map will be less or equal to the sum of the VC dimensions of each 1-dimensional components h_i if all these VC dimensions are finite; if one of the components has infinite VC dimension, then the VC dimension of the N-dimensional map will also be infinite.

Traditionally VC dimension is applied to the derivation of the necessary and sufficient conditions on the convergence of the learning algorithms. Though we would not be deriving these conditions in the current work, we can hypothesize that the proposed modification can be similarly used in necessary and sufficient conditions on the convergence of the learning algorithm with its risk functions having N-dimensional outputs.

The main purpose of the modification is that we can formally define the complexity of the combination functions which have multi-dimensional outputs as illustrated in Figure 2.2.1. Thus we define the complexity of the combination algorithm as a VC dimension of a trainable set of functions $\{s_k^j\}_{j=1,...,M;k=1,...,N} \Rightarrow \{S_i\}_{i=1,...,N}$ used in the combination.

If the one-dimensional components of the combination function have finite VC dimensions, then the total VC dimension of the combination is finite and less or equal to the sum of the components' VC dimensions. If we view the VC dimension as a number of trainable parameters (for certain functions) then such summation makes sense. In the following sections we will assume that all one-dimensional components of the combination function have the same complexity and generally these components are independently trained. Thus if C_f is the complexity of one component, then the total complexity of combination is (generally) NC_f.

2.2.2 Complexity Based Combination Types

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. Let $C_f(k)$ be the complexity of the one-dimensional component of the combination function where k is the number of input parameters. As in Figure 2.2.1 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. The complexity of this combination type is $C_f(M)$.

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. Assuming that the complexity of each combination function f_i is same $C_f(M)$ the total

complexity of combination is $NC_f(M)$.

3. Medium complexity II combinators: $S_i = f(\{s_i^j\}_{j=1,...,M}, \{s_k^j\}_{j=1,...,M;k=1,...,N,k\neq i})$. 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. The number of parameters for this function is $N * M$ and the complexity of combination is $C_f(NM)$.

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

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.

In practice,we first see if a particular classifier combination problem can be solved with high complexity combinations as a most general combination type. If complexity $NC_f(NM)$ is too big for the available training data size, number of classes N and the complexities of chosen combination functions $NC_f(NM)$, we consider lower complexity combinations. 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 as shown in Figure 2.2.2. 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.

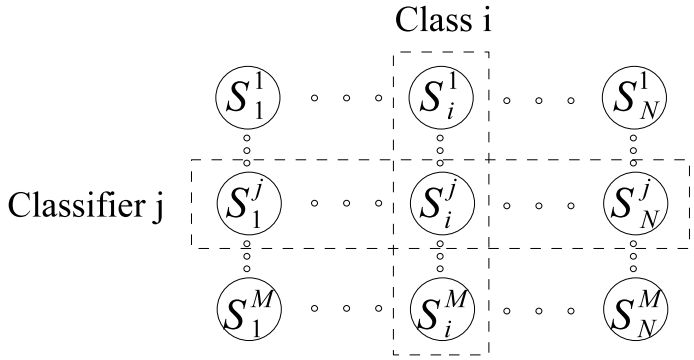
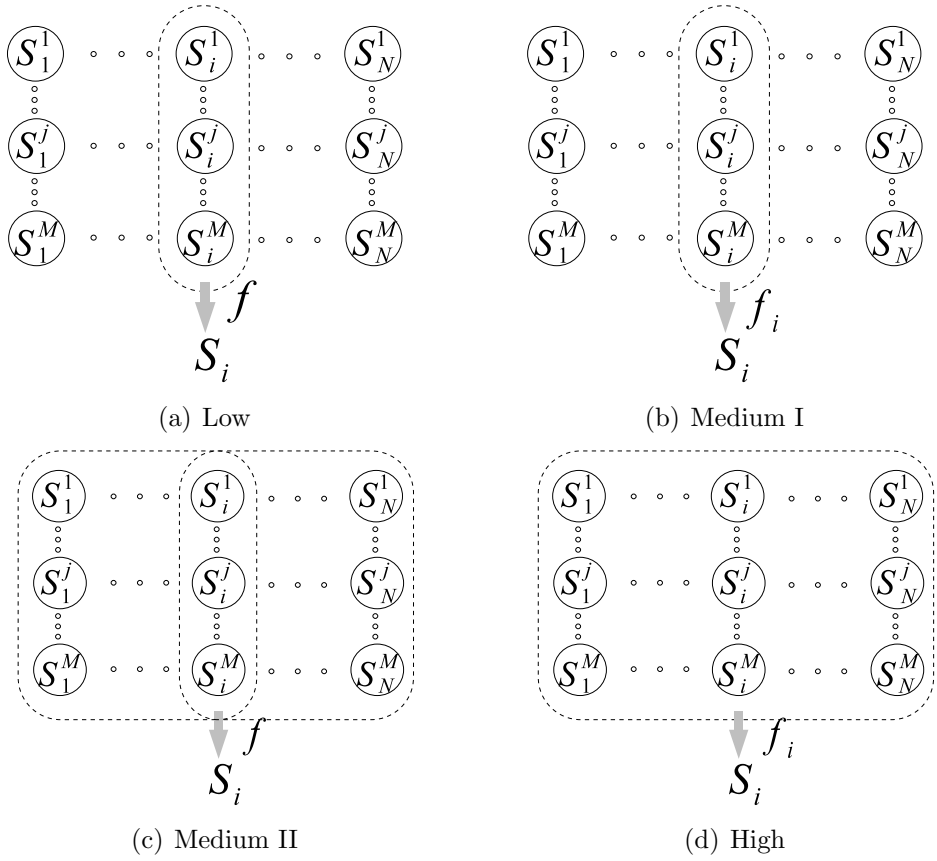


Figure 2.2.2: Output classifier scores arranged in a matrix; s_i^j - score for class i by classifier j.

The illustration of each combination type functions is given in Figure 2.2.3. In order to produce the combined score S_i for 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



(a) Low (b) Medium I

(c) Medium II (d) High

Figure 2.2.3: The range of scores considered by each combination type and combination functions.

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.

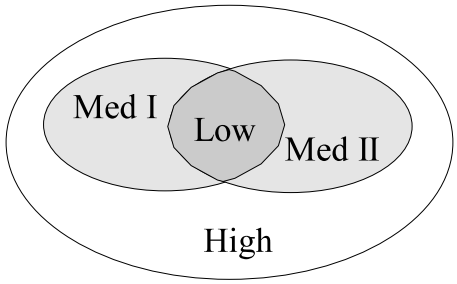


Figure 2.2.4: The relationship diagram of different combination complexity types.

Figure 2.2.4 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 Kuncheva et al.[43]. 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 [42] 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 possible participation of all scores of this classifier), and in the second step scores are combined by a function from 'low' complexity type. Thus scores in each row are combined first, and then the results are combined columnwise in the second step. This thesis explores row-wise combinations in detail in chapter 4 ('identification model'), and explores combinations using this identification model in chapter 5.

Note that Kuncheva[42] also separates nontrainable and trainable classifiers. We only consider the problem of finding the best combination algorithm for a few fixed

available classifiers. We explain in section 2.4 our rationale for not using nontrainable combination methods for this problem.

2.2.3 Solving Combination Problem

The problem of combining classifiers is essentially a classification problem in the score space $\{s_i^j\}_{j=1,...,M;i=1,...,N}$. 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?

The difference between the combination problem and the general pattern classification problem is that in 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 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. There is no theoretical basis for discarding such dependencies. As it is shown later in this thesis such dependencies do exist in real-life applications, and accounting for them can make a difference in the performance of the combination algorithm.

The classifier combination problem can be regarded as a problem of first choosing

particular complexity type for combination, then choosing an appropriate combination algorithm and at last modifying combination algorithm to account for the choice of complexity type. Usually, the set of training score vectors is the only information available for these tasks and we should make choices based on this set. The other side of the combination problem is to see if some other information besides the training score set is available and if it can be used in the combination algorithm. The meaning of the scores, though is not exact information, serves us to separate combination methods into complexity types, and can provide insights into methods to compensate for a chosen complexity type. Other information in addition to training score set can also be available for particular problems, for example independence between scores. Combination methods can utilize such information in order to justify the choice of the complexity type and, in general, for the improvement of the combination algorithm.

Below we give examples where there is a need for constructing specialized combination algorithms.

1. The number of classes N is large. This problem is quite common. Instead of considering high complexity combinations with complexities $NC_f(NM)$, medium complexity II ($C_f(NM)$) and low complexity ($C_f(M)$) combinations can be used in such situations.

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. Thus the complexity of combination is reduced in these cases by taking low complexity combination function f so that $C_f(NM)$ or $C_f(M)$ is small.

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 joint score density of M classifiers as a product of M separately estimated score densities of each classifier. The complexity type of the combination is not necessarily reduced in this case, but restriction on the set of trainable combination functions f helps to better train f.

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 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. For example, in biometric identification problems each of the N enrolled persons can have only one available training sample, thus resulting in N training samples for an MN-dimensional score space. Clearly, performing classification in such score space is not a viable option.

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.

2.3 Large Number of Classes

The situation with large number of classes arises frequently in 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 one for biometric applications where 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. High complexity combinations ($NC_f(NM)$) and medium I complexity combinations ($NC_f(M)$) might not be reliably trained because of the large multiplier N.

The remaining 2 combination types might provide a solution to the problem. The low complexity combinations are used almost exclusively for combination problems with large number of classes. However, it is possible that medium II complexity type combinations can also be used in this situation. The complexity term $C_f(NM)$ will require that combination function f is specially chosen so that $C_f(NM)$ is not big. The advantage of this combination type is that more complex relationships between classifiers' scores can be accounted for. In chapter 5 we consider this type of combination in detail.

2.4 Large Number of Classifiers

The main topic of this thesis 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 some training set. The purpose of the combination algorithm is to learn the behavior of these classifiers and produce an efficient combination

function.

Another approach to combinations includes methods trying not only to find the best combination algorithm, but also trying to find the best set of classifiers for the combination. In order to use this type of combinations there should be some method of generating a large number of classifiers. Few methods of 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[45]. A third method applies different training conditions, e.g. choosing random initial weights for neural network training or choosing dimensions for decision trees [26]. The ultimate method for generating classifiers is a random separation of feature space into the regions related to particular classes [39].

Simplest methods of combination apply some fixed functions to the outputs of all the generated classifiers (majority voting, bagging [10]). More complex methods, such as boosting [52, 20], stack generalization [62], 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 [59, 60]. But it seems that such work will be quite difficult. For example, it is noted [53] that unrestricted ensemble of classifiers has 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 classifier ensemble problem was developed by Kleinberg in the theory of stochastic discrimination[38, 39]. 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.

In our framework of combination, the complexity terms $C_f(NM)$ or $C_f(M)$ present in the different combination types, will have large values of M. Hence, we must use low complexity combination functions f in order to be able to train the combination algorithm. But if the used function f is of low complexity (for example, f is a fixed function), then complexities of all combination types can be low (especially if the number classes N is small). Thus instead of traditional low complexity combinations ($C_f(M)$) we might as well use class-specific medium I ($NC_f(M)$) and high ($NC_f(NM)$) and non-class-specific medium II ($C_f(NM)$) combinations. Using such combinations for classifier ensembles is a good topic for research.

Below we present additional discussion on ensembles of classifiers, in particular, the applicability of different combination rules. We also prove the equivalence of combination rules combining scores using symmetrical functions.

2.4.1 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[40, 57, 10]. There are few definitions of such decompositions[20].

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 typical trained classifier and an average one.

The framework of Tumer and Ghosh[58] associates trained classifiers with the approximated feature vector densities of each class. This framework has been used in many papers on classifier combination recently[41, 42, 21, 22]. In this framework, trained classifiers provide approximations to the true posterior class probabilities or to the true class densities:

$$f_i^m(x) = p_i(x) + \epsilon_i^m(x)$$

where i is the class index and m is the index of 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

$$\epsilon_i^m(x) = \beta_i(x) + \eta_i^m(x)$$

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:

$$E_{add}^m \sim \sigma^2_{\eta_i^m} + \sigma^2_{\eta_j^m}$$

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

$$E_{add}^{ave} \sim \sigma^2_{\eta_i^{ave}} + \sigma^2_{\eta_j^{ave}} = \frac{\sigma^2_{\eta_i} + \sigma^2_{\eta_j}}{M}$$

The application of the described theory is very limited in practice since too many assumptions about classifiers are required. Kuncheva[42] even compiles a list of used assumptions. Besides independence assumption of errors, we need to hypothesize about error distributions, that is the 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 [41, 3] in trying to make some 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.

2.4.2 Fixed Combination Rules for Ensembles

In this section we outline the reasons why we differentiate between the combination algorithms for fixed classifiers and combination algorithms for ensembles. The main assumption about ensembles is that all classifiers in the ensemble approximate the same posterior class probabilities $P(c_i|x)$ or same class densities $p_i(x)$, where x is a feature vector or some input unique for all ensemble classifiers and i is the index of the class. On the other hand, our research focus is on the combination methods for fixed classifiers which might have different inputs and are not supposed to approximate a single function.

The difference becomes apparent if we visualize the distributions of scores in both cases. Suppose we have two classifiers outputting scores s_i^1, s_i^2 in the range $[0, 1]$. In the case of ensemble classifiers both scores will approximate the same function f_i(posterior probability or density) and for each classification attempt with input x they will have approximately the same value: $s_i^1 \sim s_i^2 \sim f_i(x)$. Thus, in the score

space pairs (s^1, s^2) will be located near the diagonal $s^1 = s^2$ and scores s_i^1 and s_i^2 will be very strongly correlated. In the case of fixed classifiers operating on different inputs there is no such strong correlation between scores. Score pairs (s_i^1, s_i^2) could in fact have arbitrary distributions in the score space. This is especially true for independent classifiers as in biometric matchers of different modalities where s_i^1 and s_i^2 are independent random variables.

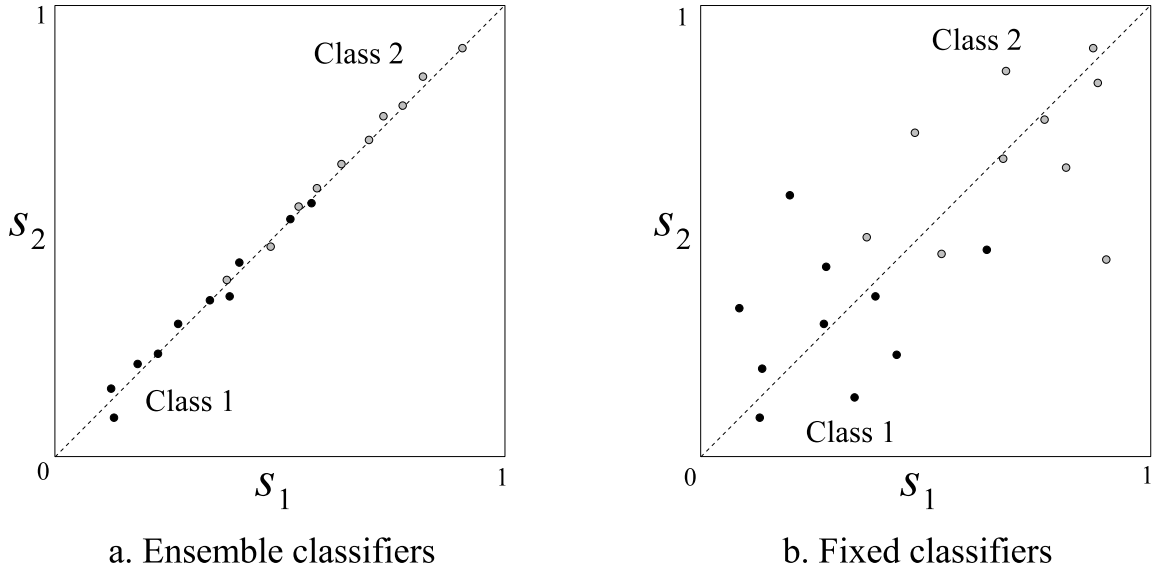
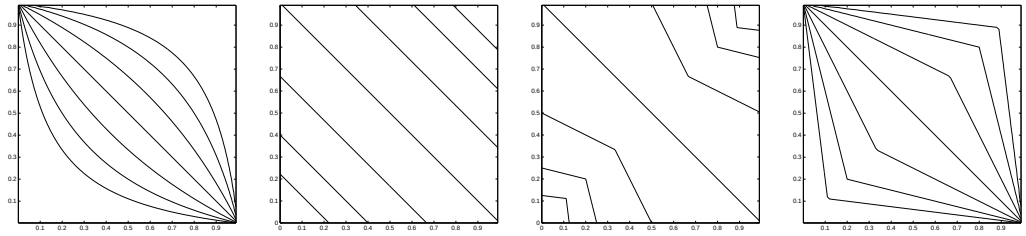


Figure 2.4.1: Sample of score distributions (s_i^1, s_i^2) of two classes.

Figure 2.4.1 shows an example of score distributions of these two cases. We assume that we have only two classes and one class has scores s_i^1 and s_i^2 close to 0 and other class has scores close to 1.

Let us now investigate how fixed combination rules separate classes in the score space. Assume we have two classes and we are combining two classifiers. Assume also that the sum of the scores assigned to two classes by each classifier equals 1: $s_1^j + s_2^j = 1$, $j = 1, 2$. This assumption holds if, for example, scores represent posterior



(a) Product rule (b) Sum rule (c) Max rule (d) Min rule

Figure 2.4.2: Fixed combination rules with ratio based decisions

class probabilities. If f is some combination rule (or rather function associated with combination rule), then $S_i = f(s_i^1, s_i^2)$, $i = 1, 2$ are combined scores assigned to two classes.

Traditionally, combination rules classify a pattern by finding the maximum of combined scores: $c_i = \arg\max_i S_i$. Let us expand this decision algorithm by basing our decision on the ratios of the combined scores: $S_1/S_2 = r$. Such decision rule is not necessarily optimal for a particular task. But if combined scores reflect posterior class probabilities then this rule would correspond to optimal Bayesian decision rule for varying prior class probabilities and different classification costs. The decision surfaces for the usual combination rules - product, sum, max and min are presented in Figure 2.4.2. Decision surfaces are drawn for $r = \{0.125, 0.25, 0.5, 1, 2, 4, 8\}$.

By comparing decision surfaces in Figure 2.4.2 with possible distributions of class samples in Figure 2.4.1 we conclude that the choice of the combination rule is not really important in the case of ensemble classifiers. In the diagonal region $s^1 \sim s^2$ where scores of ensemble classifiers are concentrated, all decision surfaces are somewhat perpendicular to the diagonal and thus make decisions in a similar manner.

The situation is quite different if we have a few fixed and not strongly correlated classifiers as in the example of Figure 2.4.1(b). Distributions of scores can be arbitrary

and the choice of combination rule can make a significant difference in the system performance. Thus using fixed combination rules to combine non-ensemble classifiers is a hit-and-miss strategy: we might get perfect decision if score distributions accidentally correspond to the chosen (or one of few tried) decision rules, or we might get a suboptimal decision. In either case we will be unable to tell if our solution is good and how far it stands apart from the optimal combination algorithm. We can summarize the above discussion in following claim:

Claim 2.1 *1. Combination of ensemble classifiers can be performed by fixed combination rules (sum, product, max, min). If all combined classifiers are trained to approximate the same function, and thus have highly correlated scores, then it does not matter which combination rule is used.*
2. In case of non-ensemble classifiers (fixed classifiers trained on different features and having non-correlated or weakly correlated scores) the use of fixed combination rules should be avoided.

These claims show that combination problems should clearly state whether they are considering classifiers ensembles or non-related classifiers. The main problem in ensemble combinations is to make sure that the generated classifiers have properties useful for combination, that is they all approximate some function with small error. And the main problem in non-ensemble combinations is finding the best combination algorithm.

2.4.3 Equivalence of Fixed Combination Rules for Ensembles

Note that the traditional case of choosing maximum of combined scores is equivalent to decision surface $r = 1$ and corresponds to the diagonal from $(0, 1)$ to $(1, 0)$ in

Figure 2.4.2 for all rules. This picture illustrates the results of previous attempts to show the equivalence of some combination rules. For example, Alexandre et al. [2] analytically prove that the product and sum rules are equivalent in the situation presented above. It would be quite trivial to prove that max and min rules are also equivalent to product and sum rule, but we omit the proof and simply reference the Figure 2.4.2.

The other point made by Alexandre[2] is that these rules are not equivalent if number of classes or number of classifiers is bigger than two. We might notice, though, that non-equivalence is proven by presenting examples of points located far away from the diagonal. Points lying outside the diagonal region pertain to the non-ensemble classifier combinations. Thus we should not consider such points when we deal with classifier ensembles.

The following theorem captures the equivalence of arbitrary combination rules in the neighborhood of the diagonal $s^1 = s^2 = \cdots = s^n$ if combination rules correspond to smooth symmetric functions.

Theorem 2.1 *Consider a classifier combination problem with 2 classes and M classifiers using notation and assumptions of the previous section. If symmetric smooth functions are used for combining scores in the combination rule, then the decision surface of the combination rule is perpendicular to the diagonal $s^1 = s^2 = \cdots = s^M$ in the score space.*

Proof: Using notations of Figure 2.2.1 $S_i = f(s_i^1, s_i^2, \ldots, s_i^M)$, $i = 1, 2$ are combined scores constructed with the help of the combination function f. The traditional combination rule requires that one finds $c_i = \arg\max_i S_i$. As in the previous section

we can consider a more general combination decision rule $D(S_1, S_2) \sim r$ (an example of $D(S_1, S_2) = S_1/S_2$ was given there), where D is some smooth function, and classification decision surfaces coincide with contours of

$$D(S_1, S_2) = D(f(s_1^1, \ldots, s_1^M), f(s_2^1, \ldots, s_2^M))$$

Now we accept assumption of $s_1^j + s_2^j = 1$, so that a set of classification scores could be identified with points lying in M-dimensional space $\{s_1^1, \ldots, s_1^M\}$. Thus the decision surfaces are contours of a function

$$F(s_1^1, \ldots, s_1^M) = D(f(s_1^1, \ldots, s_1^M), f(1 - s_1^1, \ldots, 1 - s_1^M))$$

Since f is a symmetric function in its arguments, it can be seen that function F also will be symmetric irrespective of the choice of D. F will also be smooth since f is assumed to be smooth and D is smooth by our choice. In order to prove the theorem, it is sufficient to show that the gradient of function F will be collinear to the diagonal $s^1 = s^2 = \cdots = s^n$, or equivalently, the gradient will have equal coordinates.

Take arbitrary point $(s_0^1, \ldots, s_0^M) = (s_0, \ldots, s_0)$ on the diagonal $s^1 = s^2 = \cdots = s^n$ and consider a gradient of F at this point:

$$\left(\frac{\partial F(s_1^1, \ldots, s_1^M)}{\partial s_1^1}, \ldots, \frac{\partial F(s_1^1, \ldots, s_1^M)}{\partial s_1^M} \right) \Big|_{(s_0^1, \ldots, s_0^M)}$$

Since F is symmetric, $F(s_1^1, \ldots, s_1^M) = F(s_1^i, s_1^1, \ldots, s_1^M)$ and

$$\frac{\partial F(s_1^1, \ldots, s_1^M)}{\partial s_1^i}\Big|_{(s_0, \ldots, s_0)} = \frac{\partial F(s_1^i, s_1^1, \ldots, s_1^M)}{\partial s_1^i}\Big|_{(s_0, \ldots, s_0)} = \frac{\partial F(s_1^1, \ldots, s_1^M)}{\partial s_1^1}\Big|_{(s_0, \ldots, s_0)}$$

In the last equality we did a change in notation: $s_1^i \to s_1^1, s_1^1 \to s_1^2$, etc. ■

The case with the number of classes N larger than 2 is difficult and will not yield to such simple analysis. The reason is that above proof used easily constructed decision functions D to separate the two classes. It would be impossible to create

such decision functions if the number of classes is larger than two. Still it might be possible to consider decision functions defined locally near the generalized diagonal region(the intersection of hypersurfaces $s_i^1 = s_i^2 = \cdots = s_i^M$, $i = 1, \ldots, N$), but not in the area of contention of three or more classes(say $s_{i_1}^j = s_{i_2}^j = s_{i_3}^j$, for some i_1, i_2, i_3 and all $j = 1, \ldots, M$).

If combination functions are not symmetric(weighted combination rules) or non-smooth(as min and max rules) then combination rules might result in a different performance. For example, if two classifiers have scores approximating same posterior class probability, but it is known that one classifier will have smaller score variance (and thus probably will have better performance for a particular training set), then making its weight in the weighted combination rule greater will result in the smaller variance of the combined scores, and thus result in a better combined classifier.

Chapter 3

Utilizing Independence of Classifiers

3.1 Introduction.

Traditional methodology for classifier combinations is to try a number of available combination methods and see which method performs best on the validation set [3, 12]. The assumption is that one combination method might be better on one particular problem, and another combination method might be better in different situation. It is difficult to predict which method is the best for a particular dataset, so one can try all of them and determine the best one through experiments. Though such approach is reasonable in practical situations, it is based on heuristic rather than on sound theory.

In this chapter we will address the question of how one can be sure that the best performing method is indeed the best. It is possible that all the tested methods do not include the best option. We also want to find how the experimentally chosen combination method performs compared to the theoretically optimal combination method. It is usually desirable to estimate the added error of classification - the measure between a particular classification algorithm and the optimal Bayesian classification, which assumes a knowledge of class feature densities. Recall that we argued that a combination algorithm is equivalent classification in score space (assuming only score datasets are available for training and not features). So the question about the magnitude of the added error for classifiers in the pattern classification field is also relevant for combinators. Namely, we want to estimate the difference between a particular combination algorithm and an optimal Bayesian classification using true score densities of classes.

These questions are the same general questions arising in the area of pattern classification: which classifier is better to use, and what is the difference in the performance of such classifier and optimal Bayesian classifier. Unfortunately, there are no clearcut answers to these questions so far, and the mathematical theory behind them is still in development. Similarly, it would be hard to answer these questions for the classifier combination problem. Fortunately, classifier combination problems can possess some information about the problem in addition to the set of training scores. Such additional information can provide the basis for the theoretical justification of a particular classifier combination strategy.

In this section we will explore the utilization of the classifier independence information in the combination process. As in chapter 2, we assume that classifiers output a set of scores reflecting the confidences of input belonging to the corresponding class.

Definition 3.1 *Classifiers C_{j_1} and C_{j_2} are independent if for any class i the output scores $s_i^{j_1}$ and $s_i^{j_2}$ assigned by these classifiers to the class i are independent random variables. Specifically, the joint density of the classifiers' scores is the product of the densities of the scores of individual classifiers:*

$$p(s_i^{j_1}, s_i^{j_2}) = p(s_i^{j_1}) * p(s_i^{j_2})$$

The assumption of classifiers independence is quite restrictive since the combined classifiers usually operate on the same input. Even when using completely different features for different classifiers the scores can be dependent. For example, features can be similar and thus dependent, or image quality characteristic can influence the scores of the combined classifiers. A low quality input will yield low scores for all matchers. In certain situations even classifiers operating on different inputs will have dependent scores, as in the case of using two fingers for identification (fingers will likely be both moist, both dirty or both applying same pressure to the sensor).

One recent application where independence assumption holds is the combination of biometric matchers of different modalities. In the case of multimodal biometrics the inputs to different sensors are indeed independent (for example, there is no connection of fingerprint features to face features). In this chapter we will investigate if the independence assumption can be used to improve the combination results.

Much of the effort in the classifier combination field has been devoted to dependent classifiers and most of the algorithms do not make any assumptions about classifier independence. Thus the question of constructing good combination algorithms employing independence of classifiers did not attract attention of researchers. Though independence assumption was used to justify some combination methods[37], such methods were mostly used to combine dependent classifiers. With the growth of biometric applications there is some previous work [32] where independence assumption

is used properly to combine multimodal biometric data. Our goal is to investigate how combination methods can effectively use the independence information, and what performance gains can be achieved.

3.1.1 Independence Assumption and Fixed Combination Rules

The assumption of classifier independence has been used before to justify the use of particular combination rules. For example, in the often cited work of Kittler et al.[37] the product rule is derived by assuming classifier independence. The product rule assumptions are that the scores represent the posterior class probabilities and that the classifiers are independent.

Even though we might deal with applications in which classifiers are indeed independent, as in multimodal biometric systems, the assumption of scores representing posterior class probabilities rarely holds. In fact, in our assumed biometric applications, the number of classes can be variable, and matchers usually do not account for the number of classes. If we attempt to perform some score normalization before applying combination rules, such normalization should be considered as a part of the combination algorithm. In the cited paper the experiment for combining multimodal scores used non-normalized scores, and the product rule was not the best one. Since score normalization is a necessary part for fixed combination rules and normalization is a part of training procedure, such rules simply represent a specific type of trainable classifier used for combinations. This is an additional reason that we do not consider simple combination rules in this thesis.

3.1.2 Assumption of the Classifier Error Independence

In the framework of bias and variance decompositions[58](see section 2.4.1) one frequent assumption necessary for the analysis of the combination rules is the independence of the training errors of combined classifiers $\epsilon_i^m(x)$. In this framework the outputs of the classifiers $s_i^m(x) = p_i(x) + \epsilon_i^m(x)$ approximate class densities $s_i = p_i(x)$ (or posterior class probabilities) which implies that the scores $s_i^m(x)$ are very strongly correlated for different m. At the same time errors $\epsilon_i^m(x)$ are assumed to be independent due to different training procedures, different training sets and possibly different classification methods.

It is rather difficult to ensure that such assumptions hold in practice. Indeed, during ensemble training it is common to reuse training samples from the same pool of training data. Or, if ensembles are constructed by utilizing different features in classifiers, it is common to reuse features. Even if features are not reused, the features themselves might be related. Finally, the classification methods might use similar algorithms (e.g. same kernels) which would still result in dependent classifier errors.

Thus we can not guarantee that classifiers have independent errors. Even though we can use the assumption of classifier error independence for the analysis of the performance of classifier ensembles, it seems that it would be almost impossible to utilize such independence assumption to improve the combination algorithm.

3.2 Combining independent classifiers.

We will investigate the combination of classifiers for 2-class problems. In particular, our research is motivated by the increased number of applications using biometric

verification and identification. In the verification problem an input signal is compared with the given stored signal and the closeness of the match is measured. The task of the recognition system is to classify a signal as belonging to one of the possible two classes: the class of signals originating from the same source as stored signal (genuine match), and the class of signals originating from the different source as stored signal (impostor match). In the identification problem we deal with k enrolled signals coming from k different sources and the task is to find to which of k corresponding classes the input signal belongs. Optionally, one more class is introduced to enclose all other signals not belonging to the k enrolled classes. Since the number of enrolled classes is usually large, the identification problem can be reduced to k verification problems, each verification producing a measure of match. The best matching class is chosen as the answer to the identification problem. Thus both verification and identification can be reduced to 2-class problems, and the combination algorithm can deal exclusively with two classes. In this chapter we assume that we have only two classes. In terms of the combination framework developed in previous chapter, we deal with low complexity combinations for two classes.

Even though two classes are considered, only one score for each matcher is usually available - the score between the input biometric and the stored biometric of the claimed identity. Consequently, we will assume that the output of the 2-class classifiers is 1-dimensional. For example, samples of one class might produce output scores close to 0, and samples of the other class produce scores close to 1. The set of output scores originating from n classifiers can be represented by a point in the n-dimensional score space. Assuming that for all classifiers samples of class 1 have scores close to 0, and scores of class 2 are close to 1, the score vectors in the combined n-dimensional space for two classes will be close to points $\{0, ..., 0\}$ and $\{1, ..., 1\}$. Any generic pattern classification algorithm can be used in this n-dimensional space

as a combination algorithm.

Note that this is somewhat different from the usual framework of k-class classifier combination, where k-dimensional score vectors are used and, for example, samples of class i are close to vector $\{0, ..., 1, ..., 0\}$ with only 1 at i-th place. In this case the scores for the n classifiers will be located in nk-dimensional space and the classification problem will be more difficult.

In the rest of this section we present several combination methods utilizing classifier independence. Although the experiments use only the first combination method employing score densities, we have listed other methods to provide a reference for future research and applications.

3.2.1 Combination using density functions.

Consider the combination problem with n independent 2-class classifiers. Let us denote the density function of scores produced by the j-th classifier for elements of class i as $p_{ij}(x_j)$, the joint density of scores of all classifiers for elements of class i as $p_i(\mathbf{x})$, and the prior probability of class i as P_i. Let us denote the region of n-dimensional score space being classified by the combination algorithm as elements of class i as R_i, and the cost associated with misclassifying elements of class i as λ_i. Then the total cost of misclassification in this problem is defined as $c = \lambda_1 P_1 \int_{R_2} p_1(\mathbf{x})d\mathbf{x} + \lambda_2 P_2 \int_{R_1} p_2(\mathbf{x})d\mathbf{x}$.

Since R_1 and R_2 cover the whole score space, $\int_{R_1} p_1(\mathbf{x})d\mathbf{x} + \int_{R_2} p_1(\mathbf{x})d\mathbf{x} = 1$. Thus

$$c = \lambda_1 P_1 \left(1 - \int_{R_1} p_1(\mathbf{x})d\mathbf{x}\right) + \lambda_2 P_2 \int_{R_1} p_2(\mathbf{x})d\mathbf{x}$$
$$= \lambda_1 P_1 - \int_{R_1} \left(\lambda_2 P_2 p_2(\mathbf{x}) - \lambda_1 P_1 p_1(\mathbf{x})\right) d\mathbf{x}$$

To minimize cost c, the region R_1 should be exactly the set of points $\mathbf{x}$ for which $\lambda_2 P_2 p_2(\mathbf{x}) - \lambda_1 P_1 p_1(\mathbf{x}) < 0$. Since we have independent classifiers, $p_i(\mathbf{x}) = \prod_j p_{ij}(x_j)$ and decision surfaces are described by the equation

$$f(\lambda_1, \lambda_2, \mathbf{x}) = \lambda_2 P_2 p_2(\mathbf{x}) - \lambda_1 P_1 p_1(\mathbf{x}) = \lambda_2 P_2 \prod_{j=1}^{n} p_{2j}(x_j) - \lambda_1 P_1 \prod_{j=1}^{n} p_{1j}(x_j) = 0 \quad (3.2.1)$$

To use the equation 3.2.1 for combining classifiers we need to learn $2n$ 1-dimensional probability density functions $p_{ij}(x_j)$ from the training samples.

3.2.2 Combination using posterior class probabilities.

Suppose we have posterior class probabilities $P_i(\omega_i|x_j) = \frac{p(x_j|\omega_i)P(\omega_i)}{p(x_j)} = \frac{p_{ij}(x_j)P_i}{p(x_j)}$ available for the combination algorithm. In this case

$$\begin{aligned} P(\omega_i|\mathbf{x}) &= \frac{p(\omega_i, \mathbf{x})}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\omega_1, \mathbf{x}) + p(\omega_2, \mathbf{x})} = \frac{p_i(\mathbf{x})P_i}{P_1 p_1(\mathbf{x}) + P_2 p_2(\mathbf{x})} \\ &= \frac{P_i \prod_{j=1}^{n} p_{ij}(x_j)}{P_1 \prod_{j=1}^{n} p_{1j}(x_j) + P_2 \prod_{j=1}^{n} p_{2j}(x_j)} \\ &= \frac{\left(P_i \prod_{j=1}^{n} p_{ij}(x_j)\right) / \prod_{j=1}^{n} p(x_j)}{\left(P_1 \prod_{j=1}^{n} p_{1j}(x_j) + P_2 \prod_{j=1}^{n} p_{2j}(x_j)\right) / \prod_{j=1}^{n} p(x_j)} \\ &= \frac{P_i \prod_{j=1}^{n} \frac{P_i(\omega_i|x_j)}{P_i}}{P_1 \prod_{j=1}^{n} \frac{P_1(\omega_1|x_j)}{P_1} + P_2 \prod_{j=1}^{n} \frac{P_2(\omega_2|x_j)}{P_2}} \\ &= \frac{P_i^{-(n-1)} \prod_{j=1}^{n} P_i(\omega_i|x_j)}{P_1^{-(n-1)} \prod_{j=1}^{n} P_1(\omega_1|x_j) + P_2^{-(n-1)} \prod_{j=1}^{n} P_2(\omega_2|x_j)} \end{aligned} \quad (3.2.2)$$

Using the fact that $P(\omega_i|\mathbf{x})p(\mathbf{x}) = p(\omega_i, \mathbf{x}) = P_i p_i(\mathbf{x})$ the decision surfaces of equation 3.2.1 can be rewritten as

$$f(\lambda_1, \lambda_2, \mathbf{x}) = \lambda_2 P_2 p_2(\mathbf{x}) - \lambda_1 P_1 p_1(\mathbf{x}) = \lambda_2 P(\omega_2|\mathbf{x})p(\mathbf{x}) - \lambda_1 P(\omega_1|\mathbf{x})p(\mathbf{x}) = 0 \quad (3.2.3)$$

or

$$f(\lambda_1, \lambda_2, \mathbf{x}) = \lambda_2 P_2^{-(n-1)} \prod_{j=1}^{n} P_2(\omega_2|x_j) - \lambda_1 P_1^{-(n-1)} \prod_{j=1}^{n} P_1(\omega_1|x_j) = 0 \quad (3.2.4)$$

The derived combination method is exactly the product rule presented in [37] with additional misclassification costs. Thus the product rule is the optimal combination rule for independent classifiers with output scores representing posterior class probabilities. If the output scores of classifiers do not represent posterior probabilities this rule might be suboptimal. For example, if scores represent not $P_i(\omega_i|x_j)$ but $ln(P_i(\omega_i|x_j))$ then the sum rule becomes the optimal rule. Unfortunately many published papers fail to mention what combined output scores represent or only do simplistic score normalizations, thus making any comparisons of the combination methods difficult. Note that this also applies to the outputs of neural networks trained on 0-1 values, which should be properly normalized to represent $P_i(\omega_i|x_j)$.

Since we have only two classes we can assume $P_1(\omega_1|x_j) + P_2(\omega_2|x_j) = 1$. Then, in order to use equation 3.2.4 for combination, we need to learn n 1-dimensional functions $P_1(\omega_1|x_j)$. Though this method requires less functions to learn when compared to the previous method, it might be more difficult to do implement [18]. On the other hand, it is observed[59] that solving the more general problem of estimating probability density functions and consequent derivation of posterior probabilities is disadvantageous to the direct estimations of posterior probabilities. Thus we can expect that by properly learning functions $P_1(\omega_1|x_j)$ we can produce better combination method than by using probability density functions.

3.2.3 Combination using transformation functions.

The idea of using normalizing functions has been explored in few papers on classifier combination; [4] has review and references to many such methods. We want to find a transformation of the output score of a classifier, so that some fixed combination rule (product, sum, etc.) would produce superior performance. We consider the following transformation of the scores:

$$y_j = f(x_j) = P_2(\omega_2|x_j) = \frac{P_2 p_{2j}(x_j)}{P_1 p_{1j}(x_j) + P_2 p_{2j}(x_j)} \tag{3.2.5}$$

We assume that the transformation is defined only for those x_j where p_{ij} is non-zero. According to formula 3.2.2, transformed scores $y_1, \ldots, y_n$ from n recognizers can be combined using formula

$$y = \frac{P_2^{-(n-1)} \prod_{j=1}^{n} y_j}{P_1^{-(n-1)} \prod_{j=1}^{n} (1 - y_j) + P_2^{-(n-1)} \prod_{j=1}^{n} y_j} \tag{3.2.6}$$

Since the combined score y represents posterior probability $P(\omega_2|\mathbf{x})$, the decision surfaces derived in formula 3.2.3 can be rewritten as

$$f(\lambda_1, \lambda_2, \mathbf{x}) = \lambda_2 P(\omega_2|\mathbf{x}) - \lambda_1 P(\omega_1|\mathbf{x}) = \lambda_2 y - \lambda_1 (1 - y) = 0 \tag{3.2.7}$$

Note that different score transformation formula 3.2.5 can yield different combination rules 3.2.6, for example, sum rule. Formula 3.2.6 has the advantage of being recursive, that is as recognizers are added, the already combined score can be used.

3.2.4 Modifying generic classifiers to use the independence assumption.

Any generic pattern classifier operating in the score space can be used as a combination method. In case of the verification problem, the two classes are 'genuine' and

'impostor' matches. For the identification problem, the two classes are the same, but a confidence score output is required for deciding the most probable identified class. Most generic classifiers do provide such confidence score.

We expect the generic pattern classifiers to learn the independence property from the training data. However, if we could incorporate independence information directly into these combination methods or their training, it would lead to better combination performance.

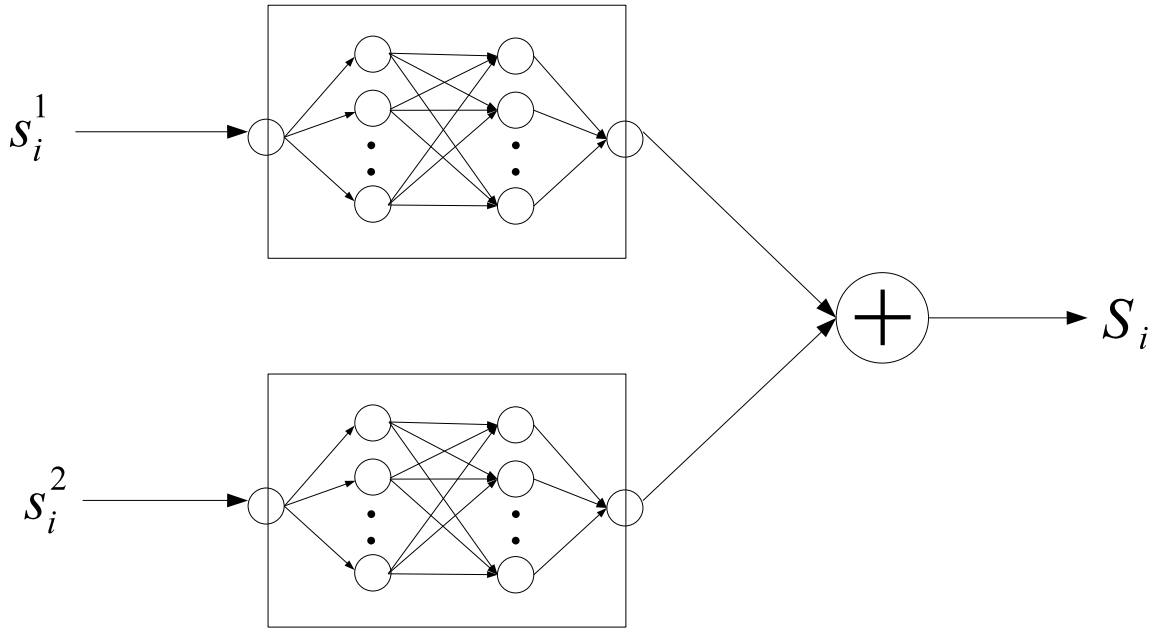


Figure 3.2.1: Possible configuration of combination neural network utilizing the independence assumption of scores s_i^1 and s_i^2.

Figure 3.2.1 presents an example of the possible modification of the neural network classifier (multilayer perceptron) used for combination to account for the independence of classifier's scores. Instead of a fully connected neural network we consider a network separately processing classifier's scores and combining them only in the last

layer. Restricting the configuration of neural network allows us to train lesser number of weights. Alternatively, we could provide more complex processing of each score. In any case we expect better performance from the modified neural network compared to a fully connected one. Similar modifications can be made on other generic classification methods such as SVMs.

3.3 Estimation of recognition error in combination

We choose the magnitude of the added error as a measure of the combination 'goodness'. It is the difference between an error of the optimal Bayesian combination and the current combination algorithm. In order to calculate this error we make a hypothesis on the true score distribution, and produce training and testing samples using these hypothesized distributions. This technique can be used to estimate the added error in practice to provide a confidence in the combination results. This would enable us to say: "'The total combination error in this particular combination is 5% and the added error due to the combination algorithm is likely to be less than 1%"'.

The added error introduced by Tumer and Ghosh[58] has been studied recently [41, 21, 22, 2]. This definition of added error assumes that the combined classifiers operate in the same feature space and the class samples are random points in this space with some fixed distributions. The Bayes error is determined by the distributions in the feature space and the added error is the difference of the combination error of trained classifiers and the Bayes error.

This framework is not applicable for combinations of biometric classifiers since these classifiers do not have the same feature space. For our task we will be treating classifiers as black boxes outputting matching scores. Scores are random variables in the

score space defined by some fixed score distributions and the combination algorithm is a classifier operating in the score space. The Baeys error, or rather the minimum Bayes cost, of the combination is determined by the score distributions. We define the combination added error, or combination added cost, as a difference between the total error(cost) and this Bayes error(cost). The difference from the previous definition is that we use distributions of scores in the score space and not the distributions of feature vectors in the feature vector space. Thus our combination added error (unlike the previously defined added error[58]) does not depend on the added errors of the individual classifiers but depends only on the combination algorithm. Section 3.3.1 gives a formal definition of combination added error.

To further explain the difference between the two types of added error, let us have an example of few imperfect classifiers operating in the same feature space. Suppose that we have the optimal combination based on the Bayesian classifier operating in the score space (assuming the score distribution are known). In this scenario the added error in Tumer and Ghosh's framework will be some non-zero number reflecting the errors made by the classifiers. By our definition the added error is 0, since the combination algorithm is perfect and did not add any errors to the classification results. Our definition is suitable for combination of fixed classifiers, whereas Tumer and Ghosh's definition works for classifier ensembles. In the next section we give a formal definition of combination added error.

3.3.1 Combination Added Error

Learning 1-dimensional probability density function $p_{ij}(x_j)$ from training samples will result in their approximations $p'_{ij}(x_j)$. Using equation 3.2.1 with these approximations will result in decision regions R'_i which ordinarily will not coincide with the optimal

Bayesian decision regions R_i. The combination added cost(AC) can be defined as a difference between the cost of using the trained regions R'_i and optimal regions R_i for combination:

$$AC = \lambda_1 P_1 \int_{R'_2} p_1(\mathbf{x})d\mathbf{x} + \lambda_2 P_2 \int_{R'_1} p_2(\mathbf{x})d\mathbf{x} \\ - \lambda_1 P_1 \int_{R_2} p_1(\mathbf{x})d\mathbf{x} - \lambda_2 P_2 \int_{R_1} p_2(\mathbf{x})d\mathbf{x} \quad (3.3.1)$$

Using set properties such as $R'_1 = (R'_1 \cap R_1) \cup (R'_1 \cap R_2)$, we get

$$AC = \int_{R'_2 \cap R_1} (\lambda_1 P_1 p_1(\mathbf{x}) - \lambda_2 P_2 p_2(\mathbf{x}))\, d\mathbf{x} \\ + \int_{R'_1 \cap R_2} (\lambda_2 P_2 p_2(\mathbf{x}) - \lambda_1 P_1 p_1(\mathbf{x}))\, d\mathbf{x} \quad (3.3.2)$$

For generic classifiers we define R'_i as the region in which samples are classified as belonging to class i. The combination added error is defined in the same way.

In order to calculate the combination added error we need to know the true distributions of the classifier's scores. We do not have this information and there is no way to calculate the combination added error. Nevertheless, we can still use the combination added error if we apply the following algorithm. First, estimate the score distributions. Second, assume that the true distributions are the same as the estimated score distributions. Finally, run simulations sampling training data sets from the assumed true distributions, training combination algorithm and comparing its performance with the assumed Bayes combination. Such simulations will provide a rough estimate on the combination added error.

In the next section we provide an example of estimating the combination added error in which we fixed the true score densities as Gaussians. Alternatively, we might have considered some approximated densities from real life classifiers in order to get the approximated combination added error.

3.4 Experiment with artificial score densiites.

In the following experiments we will assume that prior costs and probabilities of classes are equal, and use the term 'error' instead of 'cost'. We will also use relative combination added error, which will be defined as combination added error divided by the Bayesian error, and this number will be used in the tables. For example, entry of 0.1 will indicate that the combination added error is 10 times smaller than the Bayesian error.

The experiments are performed for two normally distributed classes with means at (0,0) and (1,1) and different variance values (same for both classes). It is also assumed that costs and prior probabilities of both classes are equal. The Bayesian decision boundary in this case is a straight line $x + y = 1$. Note that both the sum and product combination rules have this line as a decision surface, and combinations using these rules would give no combination added error. This is the situation where specific distributions would favor particular fixed combination rules, and this is why we eliminated these rules from our experiments.

The product of densities method described in the previous section is denoted here as DP. The kernel density estimation method with normal kernel densities [55] is used for estimating one-dimensional score densities. We chose the least-square cross-validation method for finding a smoothing parameter. Arguably, the choice of a normal kernel would favor this combination method given the underlying normal distributions. We employ kernel density estimation Matlab toolbox [6] for implementation of this method.

For comparison we used generic classifiers provided in PRTools[17] toolbox. SVM is a support vector machine with second order polynomial kernels, Parzen is a density

estimation Parzen classifier, and NN is back-propagation trained feed-forward neural net classifier with one hidden layer of 3 nodes.

Each experiment simulates sampling score distributions to obtain training data, training classifiers with this training data and evaluating classifier performance. Since score distributions are available, it is possible to generate an arbitrarily large testing set, but instead we simply use formula 3.3.2 to numerically obtain the added error. For each setting we average results of 100 simulation runs and take it as the average added error. These average added errors are reported in the tables.

In the first experiment (Table 3.4.1) we tried to see what added errors different methods of classifier combination have relative to the properties of score distributions. Thus we varied the standard deviation of the score distributions (STD) which varied the minimum Bayesian error of classifiers. All classifiers in this experiment were trained on 300 training samples.

STD	Bayesian error	DP	SVM	Parzen	NN
0.2	0.0002	1.0933	0.2019	1.2554	3.1569
0.3	0.0092	0.1399	0.0513	0.1743	0.1415
0.4	0.0385	0.0642	0.0294	0.0794	0.0648
0.5	0.0786	0.0200	0.0213	0.0515	0.0967

Table 3.4.1: Dependence of combination added error on the variance of score distributions.

The first observation is that smaller standard deviations result in larger relative added errors. This is expected in the case of density based classifiers because of the inherent difficulty of estimating density in the tails of distributions. Small standard deviation means that the optimal decision boundary is at the ends of both class distributions, and a density based method will work poorly there. Interestingly, SVM and NN classifiers also showed similar behavior. Another observation is that SVM shows

better performance than all the other methods, especially for low Bayesian error cases. Only for $STD = .5$, DP method is able to match the SVM performance.

In the second experiment (Table 3.4.1) we wanted to see the dependency of combination added error on the size of the training data. We fixed the standard deviation to be 0.5 and performed training/error evaluating simulations for 30, 100 and 300 training samples.

Number of training samples	DP	SVM	Parzen	NN
30	0.2158	0.1203	0.2053	0.1971
100	0.0621	0.0486	0.0788	0.0548
300	0.0200	0.0213	0.0515	0.0967

Table 3.4.2: Dependence of combination added error on the training size.

As expected, the added error diminishes with increased training data size. It seems that the DP method improves faster than other methods with increased training data size. Interestingly, the magnitude of the added error is relatively small for all methods. Note that we did not experiment with the number of hidden units of neural network, which might explain why its performance did not improve much with the increase of training samples.

For the third experiment (Table 3.4.3) we study how the added error changes if we combine 3 classifiers instead of 2. We take normally distributed scores with standard deviations of .4 and the size of the training data as 30. Although the additional classifier increases the relative combination added error, the significant decrease of Bayesian error would be much more important for total error. Also note that the result for 3 classifiers and results of first two rows of Table 3.4.1 have comparable Bayesian errors, with the SVM method not performing as well as for two classifiers.

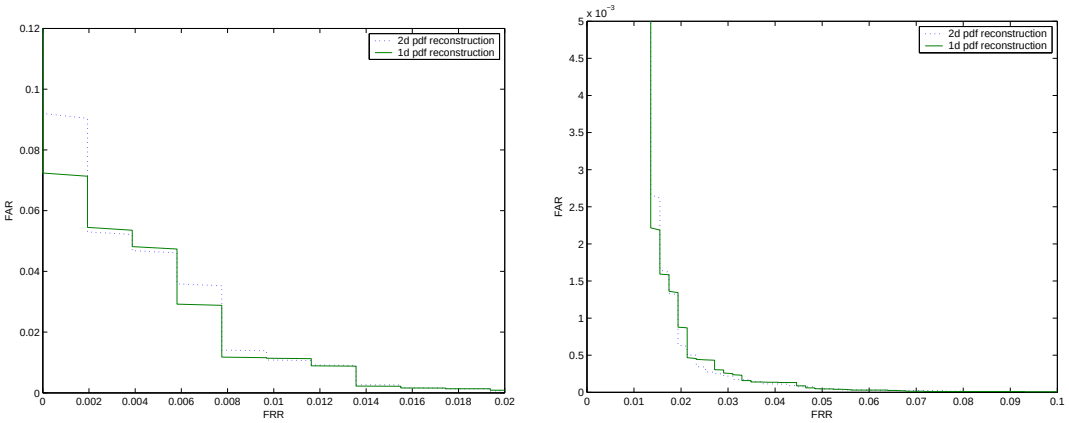
Number of classifiers	Bayes error	DP	SVM	Parzen	NN
2	0.0385	0.2812	0.1645	0.2842	0.2669
3	0.0004	0.8544	0.7882	0.6684	0.8747

Table 3.4.3: Dependence of combination added error on the number of classifiers

3.5 Experiment with biometric matching scores.

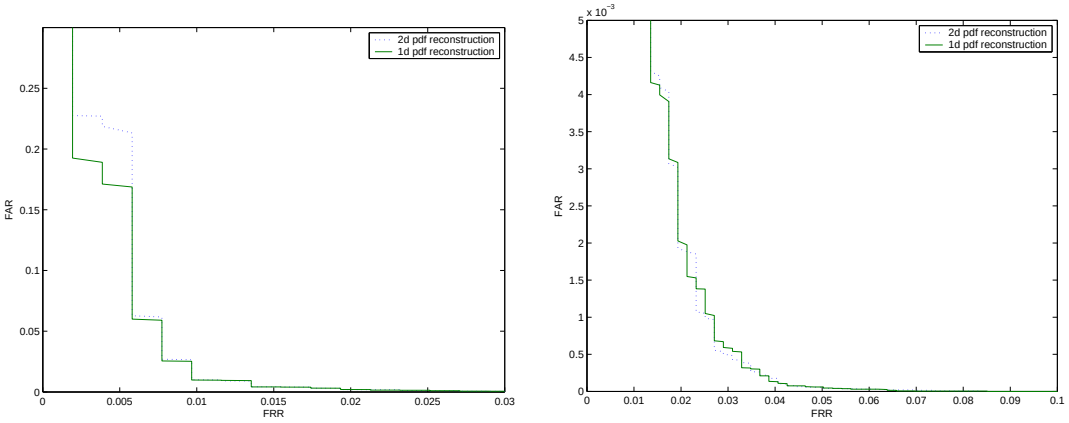
First experiment provided valuable observations on the impact of utilizing the knowledge of the score independence of two classifiers. The reported numbers are averages over 100 simulations of generating training data, training classifiers and combining them. Caution should be exercised when applying any conclusions to real life problems. The variation of performances of different combination methods over these simulations is quite large. There are many simulations where 'worse in average method' performed better than all other methods for a particular training set. Thus, in practice it is likely that the method, we find best in terms of average error, is outperformed by some other method on a particular training set. We performed experiments comparing performances of density approximation based combination algorithms (as in example 1) on biometric matching scores from BSSR1 set [1]. The results of these experiments are presented in Figures 3.5.1 and 3.5.2.

In the first figure we combine scores from the left index fingerprint matching (set li) and face (set C) matching. In the second figure we combine the same set of fingerprint scores and different set of face scores (set G). In both cases we have 517 pairs of genuine matching scores and 517*516 pairs of impostor matching scores. The experiments are conducted using leave-one-out procedure. For each user all scores for this user (one identification attempt - 1 genuine and 516 impostor scores) are left out for testing and all other scores are used for training the combination algorithm (estimating densities



(a) Low FRR range (b) Low FAR range

Figure 3.5.1: ROC curves for BSSR1 fingerprint (li set) and face (C set) score combinations utilizing and not utilizing score independence assumption.



(a) Low FRR range (b) Low FAR range

Figure 3.5.2: ROC curves for BSSR1 fingerprint (li set) and face (G set) score combinations utilizing and not utilizing score independence assumption.

of genuine and impostor matching scores). The scores of 'left out' user are then evaluated on the ratio of impostor and genuine densities providing test combination scores. All test combination scores (separately genuine and impostor) for all users are used to create the ROC curves. We use two graphs for each ROC curve in order to show more detail. The apparent 'jaggedness' of graphs is caused by individual genuine test samples - there are only 517 of them and most are in the region of low FAR and high FRR.

Graphs show we can not assert the superiority of any one combination method. Although the experiment with artificial densities shows that reconstructing one-dimensional densities and multiplying them instead of reconstructing two-dimensional densities results in better performing combination method on average, on this particular training set the performance of two methods are roughly the same.

3.6 Asymptotic properties of density reconstruction

In this section we provide a theoretical explanation for the improvement we see when we use the product of approximated one-dimensional score densities instead of reconstructing two-dimensional densities.

Let us denote true one-dimensional densities as f_1 and f_2 and their approximations by Parzen kernel method as $\hat{f}_1$ and $\hat{f}_2$. Let us denote the approximation error functions as $\epsilon_1 = \hat{f}_1 - f_1$ and $\epsilon_2 = \hat{f}_2 - f_2$. Also let f_{12}, $\hat{f}_{12}$ and ϵ_{12} denote true two-dimensional density, its approximation and approximation error: $\epsilon_{12} = \hat{f}_{12} - f_{12}$.

We will use the mean integrated squared error in current investigation:

$$MISE(\hat{f}) = E\left(\int_{-\infty}^{\infty}(\hat{f}-f)^2(x)dx\right)$$

where expectation is taken over all possible training sets resulting in approximation $\hat{f}$. It is noted in [24] that for d-dimensional density approximations by kernel methods

$$MISE(\hat{f}) \sim n^{-\frac{2p}{2p+d}}$$

where n is the number of training samples used to obtain $\hat{f}$, p is the number of derivatives of f used in kernel approximations (f should be p times differentiable), and window size of the kernel is chosen optimally to minimize $MISE(\hat{f})$.

Thus approximating density f_{12} by two-dimensional kernel method results in asymptotic MISE estimate

$$MISE(\hat{f}_{12}) \sim n^{-\frac{2p}{2p+2}}$$

But for independent classifiers the true two-dimensional density f_{12} is the product of one-dimensional densities of each score: $f_{12} = f_1 * f_2$ and our algorithm presented in the previous sections approximated f_{12} as a product of approximations of one-dimensional approximations: $\hat{f}_1 * \hat{f}_2$. MISE of this approximations can be estimated as

$$\begin{aligned} MISE(\hat{f}_1 * \hat{f}_2) = E\left(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left(\hat{f}_1(x) * \hat{f}_2(y) - f_1(x) * f_2(y)\right)^2 dxdy\right) = \\ E\left(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left((f_1(x)+\epsilon_1(x)) * (f_2(y)+\epsilon_2(y)) - f_1(x) * f_2(y)\right)^2 dxdy\right) = \\ E\left(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left(f_1(x)\epsilon_2(y) + f_2(y)\epsilon_1(x) + \epsilon_1(x)\epsilon_2(y)\right)^2 dxdy\right) = \end{aligned} \tag{3.6.1}$$

By expanding power 2 under integral we get 6 terms and evaluate each one separately below. We additionally assume that $\int_{-\infty}^{\infty} f_i^2(x)dx$ is finite, which is satisfied if, for example, f_i are bounded (f_i are true score density functions). Also, note that

$$MISE(\hat{f}_i) = E\left(\int_{-\infty}^{\infty}(\hat{f}_i - f_i)^2(x)dx\right) = E\left(\int_{-\infty}^{\infty}(\epsilon_i)^2(x)dx\right) \sim n^{-\frac{2p}{2p+1}}.$$

$$E\left(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_1^2(x)\epsilon_2^2(y)dxdy\right) = \int_{-\infty}^{\infty} f_1^2(x)dx * E\left(\int_{-\infty}^{\infty}\epsilon_2^2(y)dy\right) \sim n^{-\frac{2p}{2p+1}}$$

$$E\left(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_2^2(y)\epsilon_1^2(x)dxdy\right) = \int_{-\infty}^{\infty} f_2^2(y)dy * E\left(\int_{-\infty}^{\infty}\epsilon_1^2(x)dx\right) \sim n^{-\frac{2p}{2p+1}}$$

$$E\left(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_1(x)\epsilon_1(x)f_2(y)\epsilon_2(y)dxdy\right) =$$

$$E\left(\int_{-\infty}^{\infty} f_1(x)\epsilon_1(x)dx\right) * E\left(\int_{-\infty}^{\infty} f_2(y)\epsilon_2(y)dy\right) \leq$$

$$\sqrt{\int_{-\infty}^{\infty} f_1^2(x)dx}\sqrt{E\left(\int_{-\infty}^{\infty}\epsilon_1^2(x)dx\right)}\sqrt{\int_{-\infty}^{\infty} f_2^2(y)dy}\sqrt{E\left(\int_{-\infty}^{\infty}\epsilon_2^2(y)dy\right)} \sim$$

$$\sim \sqrt{n^{-\frac{2p}{2p+1}}}\sqrt{n^{-\frac{2p}{2p+1}}} = n^{-\frac{2p}{2p+1}}$$

$$E\left(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_1(x)\epsilon_1(x)\epsilon_2^2(y)dxdy\right) =$$

$$E\left(\int_{-\infty}^{\infty} f_1(x)\epsilon_1(x)dx\right) * E\left(\int_{-\infty}^{\infty}\epsilon_2^2(y)dy\right) \leq$$

$$\sqrt{\int_{-\infty}^{\infty} f_1^2(x)dx}\sqrt{E\left(\int_{-\infty}^{\infty}\epsilon_1^2(x)dx\right)}E\left(\int_{-\infty}^{\infty}\epsilon_2^2(y)dy\right)$$

$$\sim \sqrt{n^{-\frac{2p}{2p+1}}}n^{-\frac{2p}{2p+1}} = o\big(n^{-\frac{2p}{2p+1}}\big)$$

Similarly,

$$E\left(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\epsilon_1^2(x)f_1(x)\epsilon_2(y)dxdy\right) = o\big(n^{-\frac{2p}{2p+1}}\big)$$

$$E\left(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\epsilon_1^2(x)\epsilon_2^2(y)dxdy\right) = E\left(\int_{-\infty}^{\infty}\epsilon_1^2(x)dx\right)E\left(\int_{-\infty}^{\infty}\epsilon_2^2(y)dy\right) = o\big(n^{-\frac{2p}{2p+1}}\big)$$

Thus we proved the following theorem:

Theorem 3.1 *If score densities of two independent classifiers f_1 and f_2 are p times differentiable and bounded, then the mean integrated squared error of their product approximation obtained by means of product of their separate approximations*

$MISE(\hat{f}_1 * \hat{f}_2) \sim n^{-\frac{2p}{2p+1}}$, whereas mean integrated squared error of their product approximation obtained by direct approximation of two-dimensional density $f_{12}(x, y) = f_1(x) * f_2(y)$ $MISE(\hat{f}_{12}) \sim n^{-\frac{2p}{2p+2}}$.

Since asymptotically $n^{-\frac{2p}{2p+1}} < n^{-\frac{2p}{2p+2}}$, the theorem states that under specified conditions it is more beneficial to approximate one-dimensional densities for independent classifiers and use a product of approximations, instead of approximating two or more dimensional joint density by multi-dimensional kernels. This theorem partly explains our experimental results, where we show that DP method (density product) of classifier combination is superior to multi-dimensional Parzen kernel method of classifier combination. This theorem applies only to independent classifiers, where knowledge of independence is supplied separately from the training samples.

3.7 Conclusion

In this chapter we evaluated combination added error. We showed experimentally that this error is relatively small for all combination methods. So it does not really matter which combination method is used to combine results of classifiers. By using a larger number of training samples, an inferior combinator can outperform a superior combinator. Thus it is more important to see what is the minimum Bayesian error of combined system, which is determined by the classifiers' performances and their interdependencies (assuming that trainable generic classifier is used as combinator and not fixed combination rules, like sum or product rule). The choice of the combination algorithm becomes more important when classifiers have small Bayesian errors.

Our method for combining independent classifiers by multiplying one-dimensional

densities shows slightly better performance than a comparable Parzen classifier. Thus using the independence information can be beneficial for density based classifiers. However, the DP method is not as good as SVM. It seems that if more training samples are used and more classifiers are combined, DP might prove to be better than SVM. Incorporating the knowledge about independence of the combined classifiers into other methods (such as neural networks or SVMs) can improve their performance as well.

The main motivation of this chapter was to find the best combination method for multimodal biometric matchers. The presented techniques should help to choose combination method, although other factors should also be taken into consideration. For example, if costs of incorrect classification or prior probabilities of classes change, the SVM or neural network method will require retraining. Also, if output of combination confidence is required for system operation, the ability of density based methods to output posterior class probability can be a decisive factor for their adoption.

Chapter 4

Identification Model

4.1 Introduction

The identification problem of pattern recognition can be loosely defined as a classification problem with variable classes. One of the applications is identifying the person using speech, handwriting or other biometric samples among n enrolled persons. The number of enrolled persons and their identities change arbitrarily, thus class relationships are frequently discarded. Other applications which we will consider in this chapter are handwriting recognition with variable lexicon, and barcode recognition.

The class relationships are usually discarded due to problem intractability. For example, the number of persons or handwritten words can be large, and the number of samples available for training can be just 1 - for biometric templates, or none - for word recognition. Therefore, most recognition algorithms do not include relationships between classes in their training procedure. Consequently, during the recognition stage

classes are matched one at a time. The class whose matching produces the best score is declared as winner.

The problem occurs it is needed to ensure that the winner class is indeed the truth for the input sample. A quick solution is to set a threshold θ and accept the winner only if the matching score is better than threshold. However, if there is a second candidate with a score close to the best score there is a fair chance that the true class might be the one with the second best score instead of first. Thus instead of condition $s_1 > \theta$, where s_1 is the best score and $>$ means better score, we might want to use condition $s_1 > \theta_1$ and $s_1 - s_2 > \theta_2$, where s_2 is the second best score. The second condition takes into consideration not only the individual matching scores, but also relations between the scores of the different classes. Thus this approach incorporates class relationship information into the classification task.

We can address several questions arising with regards to the above described problem:

1. How are thresholds θ_1 and θ_2 determined?
2. Is there a better way to combine s_1 and $s_1 - s_2$ to obtain the classification confidence?
3. Can we use $s_3, s_4, \ldots$ to improve confidence estimates?
4. How do the problem parameters (e.g. number of classes) influence the performance gained by using interclass score relationships?

Expanding identification decision to include the second best score is intuitive and commonly used. Brunelli and Falavigna[11] use the ratio of the normalized best and second best scores in the decision on person identification based on speech and facial features. Another recent paper[54] uses the best score and average of next N scores

to decide on writer identification. However, a theoretical properties of the use of the second-best scores have not been adequately studied. Grother and Phillips[23] use a simple threshold condition of the first type while building the model of identification performance.

The related idea of combining matching scores is to somehow use local neighborhood information of the matched pattern. For example, the technique of score normalization in speaker verification [50, 15, 47] uses matching scores of a set of similar speakers (cohort) to improve the verification confidence. Also there are works where this technique is successfully applied for speaker identification[35]. Using background models[50] for speaker verification implies the use of distribution of non-matching classes, and the normalized score rather approximates optimal identification score - a Bayes posterior probability of matching input to particular class. This implicit assumption about non-matching score distribution might explain the difficulty in deriving strict a mathematical framework for score normalization in the verification problem.

In this chapter we will use artificial examples to illustrate the effects of using second-best scores in the identification process. We will also give examples of using second-best scores in real life applications.

4.2 Combining Matching Scores under Independence Assumption

Let N be the number of classes in our identification problem, $s_1 > s_2 > \cdots > s_N$ are the scores produced during an identification attempt (index shows the order of

scores and not class number). One of the scores s_i is produced by matching with truth class, $id(input) = id(i)$. Let p_m and p_n denote the densities of of matching and non-matching scores. We fix the densities for our example as

$$p_m(s) = c_m e^{-\frac{(1-s)^2}{2\sigma_m^2}}$$

$$p_n(s) = c_n e^{-\frac{s^2}{2\sigma_n^2}}$$

where $s \in [0,1], \sigma_m = .4, \sigma_n = .2$ and c_m and c_n are normalizing constants. The densities for matching and non-matching scores are shown in Figure 4.2.1.

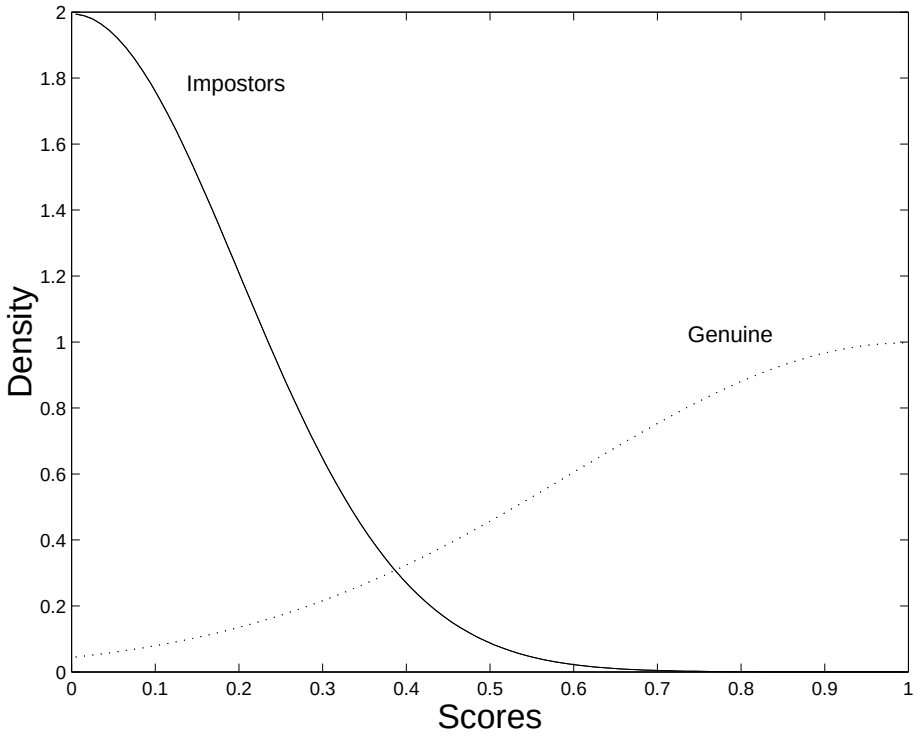


Figure 4.2.1: Chosen densities of matching(genuine) and non-matching(impostors) scores.

The main assumption which we make in this example is that the scores s_i produced during matching input pattern against all classes are independent random variables.

One score from the truth class is sampled from p_m and the remaining $N-1$ scores are sampled from p_n. This assumption is rather restrictive and generally it is not true. For example, frequently matching score includes some measure of input signal quality. Since the quality of input is the same for all matching attempts, we expect that scores $s_1, s_2, \ldots, s_N$ will be dependent.

By using the independence assumption we are able to calculate the joint density of best and second-best scores under two conditions: best score comes from matching truth class and best score comes from matching non-truth class. These are the formulas for densities in case $N = 2$:

$$\begin{aligned} p(s_1, s_2|id(input) = id(1)) &= p_m(s_1) * p_n(s_2) \\ p(s_1, s_2|id(input) \neq id(1)) &= p_n(s_1) * p_m(s_2) \end{aligned} \tag{4.2.1}$$

Bayes decision theory holds that optimal decision surfaces are defined by the likelihood ratio:

$$L = \frac{p(s_1, s_2|id(input) = id(1))}{p(s_1, s_2|id(input) \neq id(1))} = \frac{p_m(s_1) * p_n(s_2)}{p_n(s_1) * p_m(s_2)} \tag{4.2.2}$$

Sample decision surfaces are shown in Figure 4.2.2. Note that if we use only the best score s_1 for making decisions, we will get vertical lines as decision boundaries. Thus decisions involving second best score substantially differ from decisions based solely on s_1. In Figure 4.2.3 we show the ROC curves for decisions utilizing the second-best score. Good reduction in false acceptance rate (FAR) is achieved when false reject rate (FRR) is around $0.1 - 0.4$. Second-best score is a good feature for making decision about the following two cases: a) best matched class is the truth, b) best matched class is not a truth.

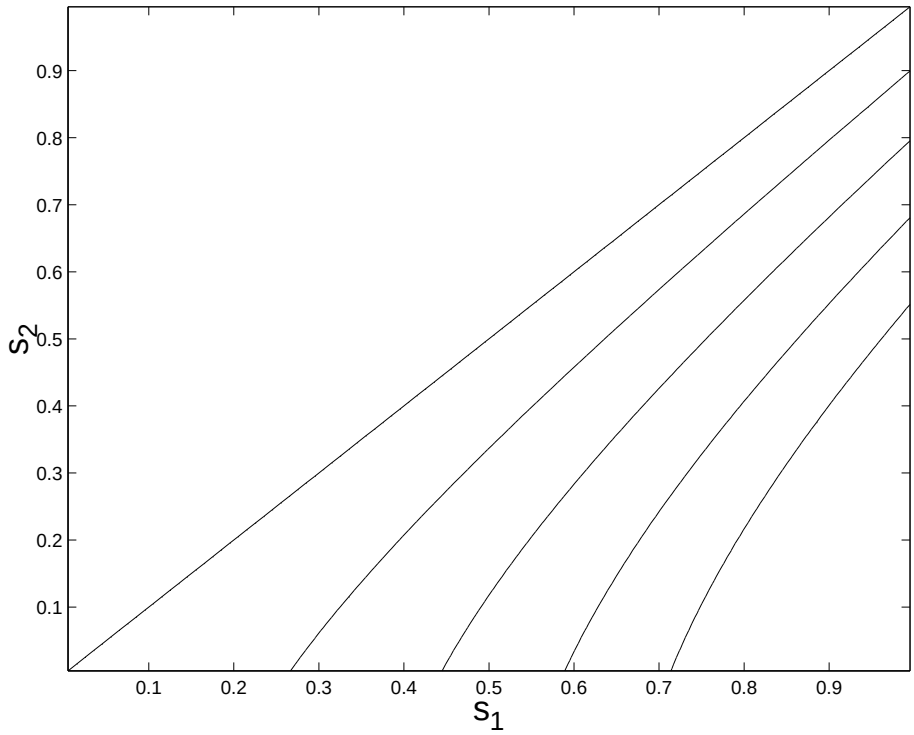


Figure 4.2.2: Bayes decision boundaries for N=2 with contours drawn for L=1,10,100,1000 and 10000.

4.3 Dependent Scores

The main assumption used in the previous example is the independence of matching scores. But as we noted this is rarely the case in practice. Is it still beneficial to use second-best score for making identification decision if scores are dependent?

Using the second-best score in addition to the best score amounts to simply adding one more feature in the two-class classification problem. In the Bayes framework which we consider in the artificial example adding more features can not make the performance worse. Thus it is possible to only improve results by considering the

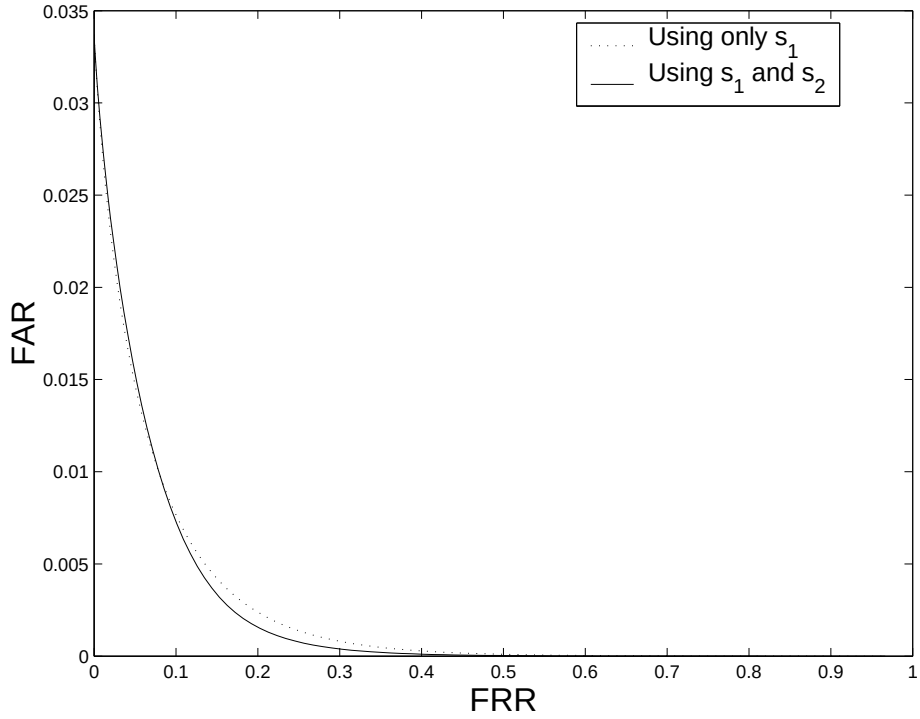


Figure 4.2.3: ROC curves for optimal thresholding using and not using second-best score.

second-best score, as well as all other scores. In practice, using the second-best or other scores depends on the number of training samples. We can identify two extreme cases of score dependency with regard to using the second-best score.

1. The worst case is where no improvement can be achieved. Consider a matcher producing scores normalized to posterior probabilities of participating classes, $s_i = P(\omega_i|input)$. The score for best class is exactly the probability that this class is genuine, and thresholding such probability coincides with the Bayesian cost minimization. Specifically, the optimal Bayesian decision boundary is determined by the ratio of genuine and impostor densities, and the posterior class

probability can be expressed through such ratio. Hence, the curve corresponding to constant posterior probability coincides with Bayesian decision boundary. Thus it is sufficient to consider only the first score for optimal decision. For $N = 2$ the scores will be connected by the equation $s_2 = 1 - s_1$, and later discussion will alternatively prove that for such scores considering second best score does not benefit decision process.

2. The best case is where combination of s_1 and s_2 achieves perfect separation of successful and unsuccessful identification events. For example, suppose the correct identifications (the best score belongs to genuine class) always have $s_1 - s_2 > .1$, and incorrect identifications have $s_1 - s_2 < .1$. In this situation, taking $s_1 - s_2 = .1$ as decision boundary will give us FAR=0 and FRR=0. Using only score s_1 or independence assumption as in formula 4.2.1 will fail to give such results.

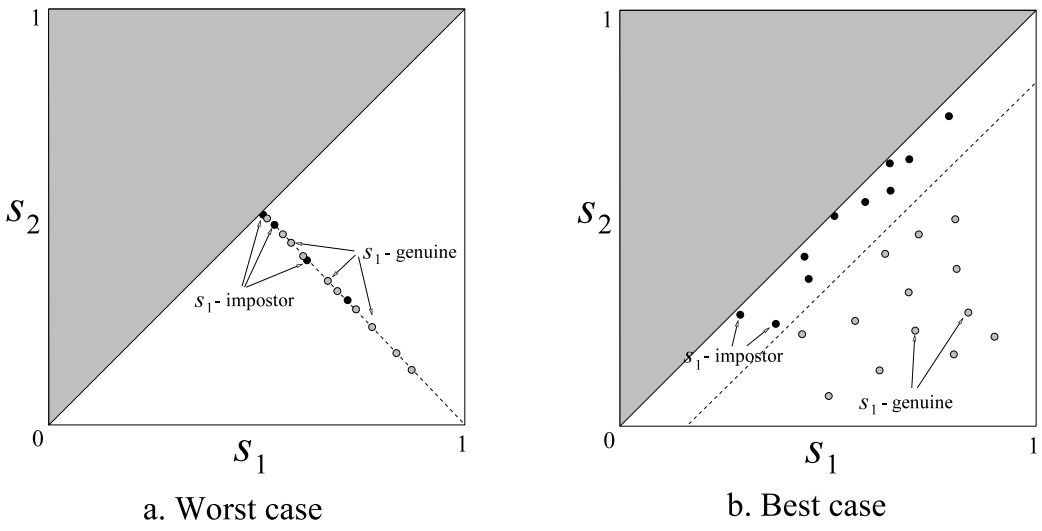


Figure 4.3.1: Two extreme cases for utilizing second best score s_2 for decision thresholding.

Figure 4.3.1 illustrates these two cases. In the first case, all scores lie on the line $s_2 = 1 - s_1$ (note that we consider only region $s_1 \geq s_2$). If we use traditional thresholding of the score s_1 only, the decision surfaces will be vertical lines. If we use any other thresholding method, the decision surfaces will still somehow intersect this line. Thus such thresholding will also produce a division of this line into two sets: 'accept' and 'reject'. Clearly, any such division could be be produced by vertical decision surfaces and by decision methods involving only the first score s_1. Thus incorporating the second score s_2 into the decision algorithm will not give any additional benefits.

In the second case line $s_1 - s_2 = .1$ will give a perfect decision separating identification attempts with the first score s_1 corresponding to the genuine class, and identification attempts with the first score corresponding to the impostor class. The case with independent scores will be similar to Figure 4.2.3, and utilizing the second best score will give some improvement during the decision step.

Scores are frequently dependent. Scores were dependent to some extent in all the applications we experimented with. We can point out at least two causes of such dependence:

1. Recognizers usually incorporate some measure of input quality into matching score. If quality of the input is low we can expect all matching score to be low, and if quality is high, then matching scores also will be high.

2. We expect character images to belong to a definite set of classes, and if a character is in one class, it will be quite different from characters in other classes. When the distortion is small and the correct class is matched, the distance to other classes will dictate low impostor scores. But if impostor class is matched, the input sample probably lies somewhere in between classes, and the second best score is comparable to the first one.

Summarizing the above discussion, if matching scores are independent we expect to achieve average performance improvement by combining the second-best score. If scores are dependent, then any situation from no improvement to perfect decision is possible. Scores are usually dependent and therefore considering second-best score in decision is beneficial.

4.4 Different number of classes N

The interesting question for identification systems is what will be the performance of the system for large number of classes, or, in the case of biometric identification systems, a large number of enrolled persons. Previous research in this area has relied on decision processes considering only the single best score [61, 23, 8]. It is assumed that the matching scores assigned to different classes during the identification attempt are statistically independent. Let FRR_1 and FAR_1 denote the false reject and false accept rates for verification system where the decision to accept is determined by some threshold θ:

$$FRR_1(\theta) = \int_{-\infty}^{\theta} p_g(s)ds \quad , \quad FAR_1(\theta) = \int_{\theta}^{\infty} p_i(s)ds$$

where $p_g(s)$ and $p_i(s)$ are densities of genuine and impostor scores. Note, that we assume (as in section 4.2) that the genuine scores are usually greater than impostor scores, and verification is accepted if $s \geq \theta$ and rejected if $s < \theta$. Suppose we have N enrolled persons and during an identification attempt we have one genuine score and $N-1$ impostor scores. The probability of false reject in the identification trial is $FRR_N(\theta) = FRR_1(\theta)$ and the probability that at least one of $N-1$ will be accepted

is

$$\begin{aligned} FAR_N(\theta) &= P(\max s_i \geq \theta, i = 1, \ldots, N-1, s_i \text{ - impostor scores}) \\ &= 1 - P(\max s_i < \theta, i = 1, \ldots, N-1, s_i \text{ - impostor scores}) \\ &= 1 - \prod_{i=1,\ldots,N-1} P(s_i < \theta, s_i \text{ - impostor scores}) \\ &= 1 - \prod_{i=1,\ldots,N-1} (1 - FAR_1(\theta)) = 1 - (1 - FAR_1(\theta))^{N-1} \end{aligned} \tag{4.4.1}$$

The usual assumptions on this model are that scores are independent and impostor scores have identical distributions. There is also an implicit assumption that the threshold defining false accept and false reject rates is defined by using a single score. For example, Wayman [61] considers different scenarios of decision policy, but all scenarios have comparisons of individual scores to some thresholds. As we showed in section 4.2, such type of thresholding is suboptimal, and we have to consider more complex comparisons, e.g. a comparison of a linear combination of the first two scores to the threshold. In this section we will consider more complex type of decision making involving a combination of scores.

We will assume that the scores s_i produced during matching the input pattern against all classes are independent random variables. One score from the truth class is sampled from p_g and the remaining $N-1$ scores are sampled from p_i. This assumption is rather restrictive and generally not true, but it will be helpful for experiments we conduct here. Using the independence assumptions we are able to calculate the joint density of the best and second-best scores for the two classes: a class where best score comes from genuine match and a class where best score comes from impostor match. Let us denote these densities as $p_{gen}(s_1, s_2)$ and $p_{imp}(s_1, s_2)$, where s_1 is the

best score, and s_2 is the second best score. Then

$$\begin{aligned} p_{gen}(s_1, s_2) =& p_g(s_1)p_i(s_2)F^i_{N-2}(s_2)(N-1) \\ p_{imp}(s_1, s_2) =& p_i(s_1)p_i(s_2)F^i_{N-3}(s_2)(N-2)(N-1)F^g(s_2) \\ &+ p_i(s_1) * p_g(s_2) * F^i_{N-2}(s_2) * (N-1) \end{aligned} \tag{4.4.2}$$

where $F^i_n(s)$ denote probabilities of n impostor scores be less than s : $F^i_n(s) = \left(\int_{-\infty}^{s} p_i(t)dt\right)^n$, and $F^g(s)$ denotes the probability of a genuine score to be less than s: $F^g(s) = \int_0^s p_g(t)dt$. We do not provide strict derivation of these formulas, but instead show how they can be interpreted. In the first formula the best score is sampled from the genuine density and thus there is the term $p_g(s_1)$. The second score is the impostor score and hence we have term $p_i(s_2)$. All other $N-2$ impostor scores are less than s_2, so there is a term $F^i_{N-2}(s_2)$. Finally, term $(N-1)$ comes from $N-1$ possibilities of the best impostor score s_2 to be from $N-1$ different impostor classes. The second formula could be explained similarly. Note that in this formula the probability of impostor match having the best score, $p_{imp}(s_1, s_2)$, is the sum of probabilities of two events: second match is the impostor match and second match is the genuine match, each corresponding to two sum terms.

Formulas 4.4.2 allow us to calculate the joint best and second-best score densities $p_{gen}(s_1, s_2)$ and $p_{imp}(s_1, s_2)$ for successful (genuine score is on top) and unsuccessful (impostor score is on top) identification attempts if the densities of the genuine and impostor scores are known. In order to investigate the performance of identification systems for different number of enrolled persons N we again consider an artificial example where $p_g(s) = c_g e^{-\frac{(1-s)^2}{2\sigma_g^2}}$ and $p_i(s) = c_i e^{-\frac{s^2}{2\sigma_i^2}}$, $s \in [0, 1], \sigma_g = .3, \sigma_i = .2$ and c_g and c_i are normalizing constants. Figures 4.4.1 through 4.4.6 present the results of these experiments.

Note that in these experiments we fixed the densities of the considered scores, and

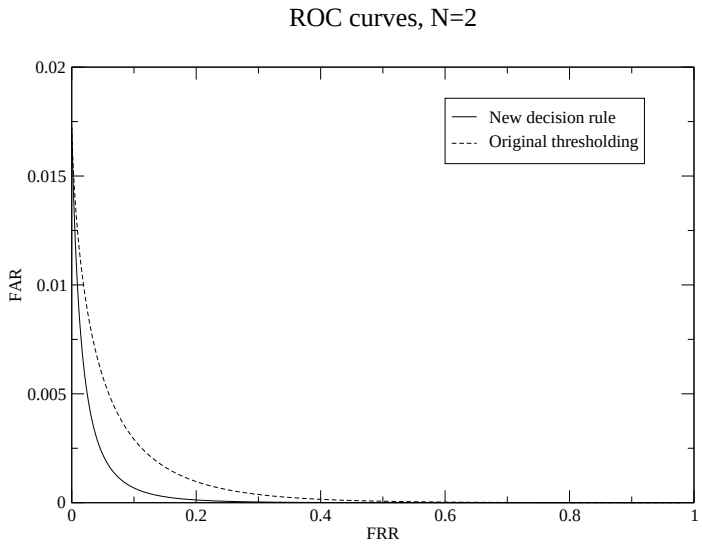


Figure 4.4.1: ROC curves for optimal thresholding using and not using second-best score, N=2.

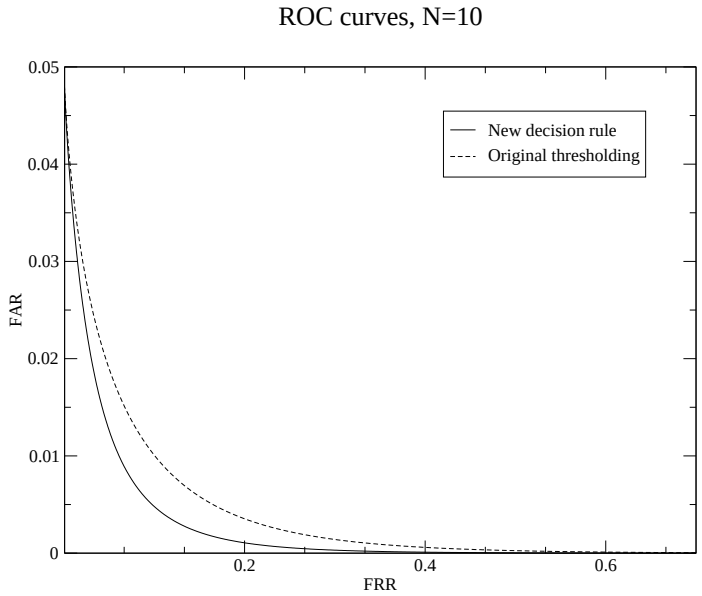


Figure 4.4.2: ROC curves for optimal thresholding using and not using second-best score, N=10.

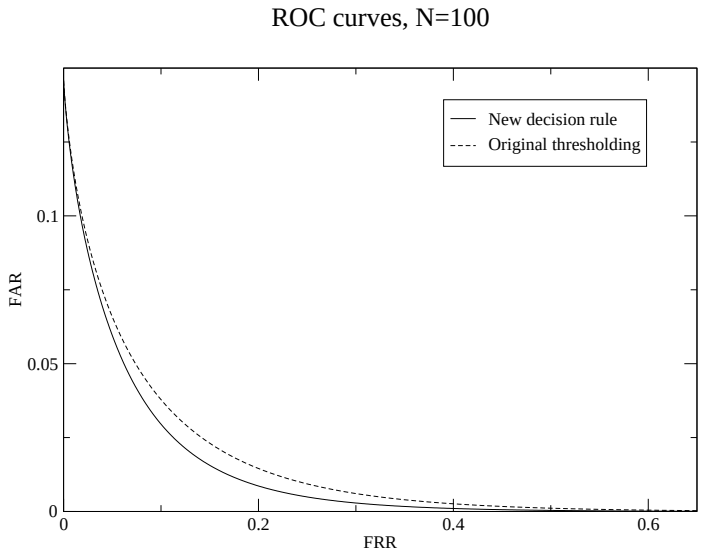


Figure 4.4.3: ROC curves for optimal thresholding using and not using second-best score, N=100.

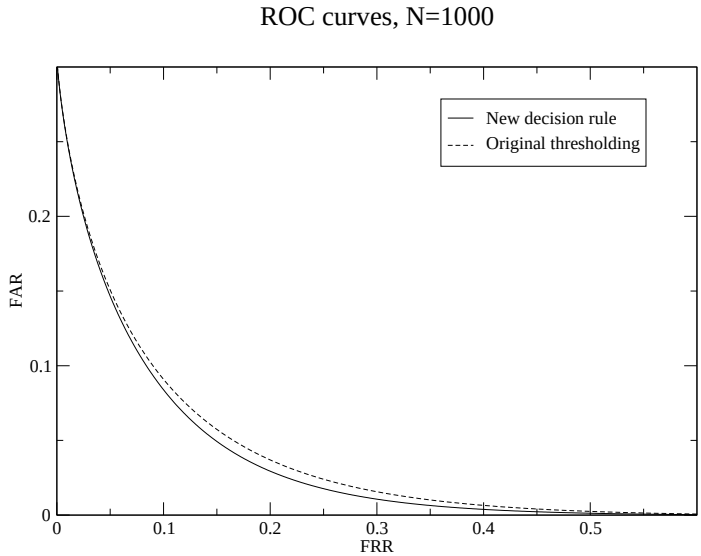


Figure 4.4.4: ROC curves for optimal thresholding using and not using second-best score, N=1000.

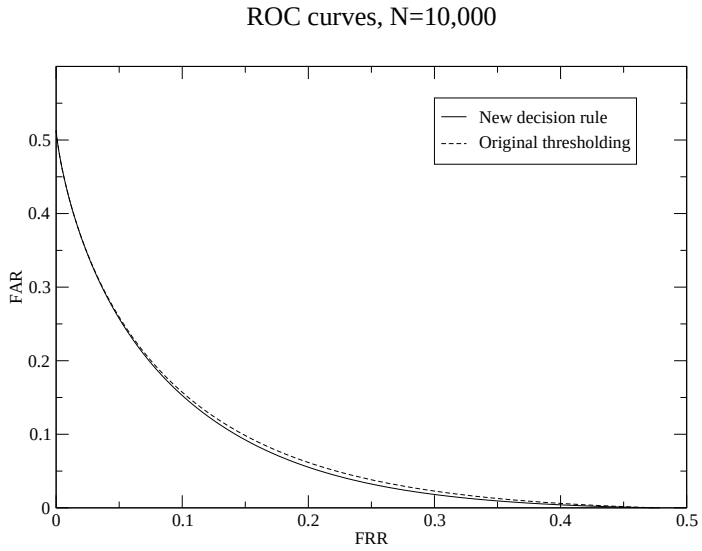


Figure 4.4.5: ROC curves for optimal thresholding using and not using second-best score, N=10,000.

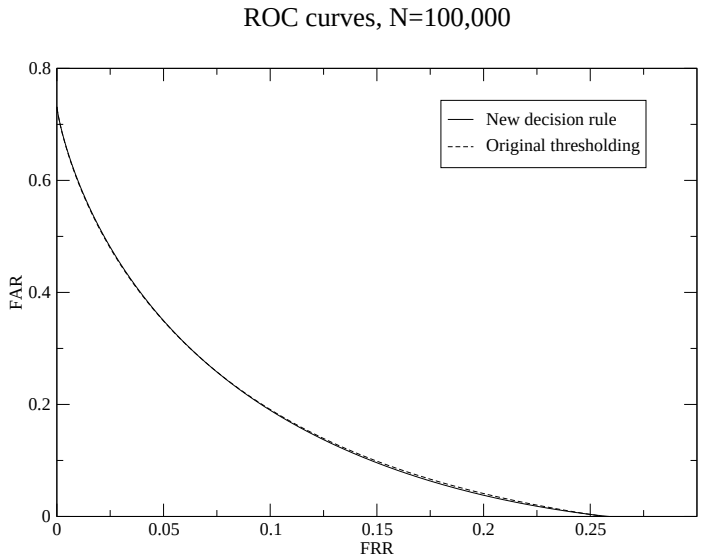


Figure 4.4.6: ROC curves for optimal thresholding using and not using second-best score, N=100,000.

formulas 4.4.2 allow direct calculations of joint densities for different number of enrolled classes N. Thus the presented results are exact and not statistical for assumed densities and independence of scores.

As it is to be expected, larger number of enrolled persons results in worse system performance: ROC curves are higher for bigger N. The benefit of using the second-best score for acceptance decision is greater when the number of enrolled persons N is small. If the number of enrollees increases, the benefit of using the second-best score diminishes. Such trend is not necessarily true for matchers with dependent scores. The next section presents an example of barcode recognition application where the number of classes is 2^{106}. Utilizing the second-best score is quite beneficial as it almost halves the error rate.

4.5 Examples of Using Second-Best Matching Scores

In this section we present examples of incorporating the second-best matching score for the identification problem in different real life applications. We will explore the problems of identifying a handwritten word given a lexicon, barcode recognition and biometric person identification.

4.5.1 Handwritten word recognition

We consider the recognition of handwritten street names in the address recognition systems. The handwritten destination address is first automatically segmented and recognition of zip codes and house numbers is performed. Based on these two numbers the database with street names is queried, and results of the query are used

as the lexicon for the word recognizer [56]. The lexicon size varies from one to few hundred phrases. The identification of street name by word recognizer serves as a decision for accepting mail piece recognition. The truth phrase of the recognized image is not always contained in the lexicon. Thus we are truly dealing with a mixture of identification and verification. Nevertheless, assuming some probability that this automatically generated lexicon contains truth, this reduces to the identification problem. The word recognizer provides matching distances as output scores. The score assigned to a particular class does not change when the lexicon is changed (but still includes this particular class). Thus scores are not normalized to posterior class probabilities, and incorporating the second-best score fits this application well.

We use logistic regression on (s_1, s_2) samples to incorporate second-best score and to model posterior probability

$$P(\text{correct identification}|s_1, s_2) \quad \sim \quad L(s_1, s_2) \quad = \quad \frac{1}{1 + e^{as_1+bs_2+c}} \quad (4.5.1)$$

We use total 7615 samples for both training (finding best values of a, b, c) and testing. 2427 samples are incorrectly identified as the best score does not correspond to the truth class.

Figure 4.5.1 shows the ROC curves based on thresholding using only the first score s_1 and on thresholding the value of the logistic function. We can see from the graph that significant reduction in false acceptance rate can be achieved for (1-FRR) between 0.2 and 0.7.

Note that the this application has variable number of classes. As we showed in section 4.4 the distribution of the best impostor score changes with N and results in different thresholding parameters. Thus it makes sense to use different decisions for different values of N.

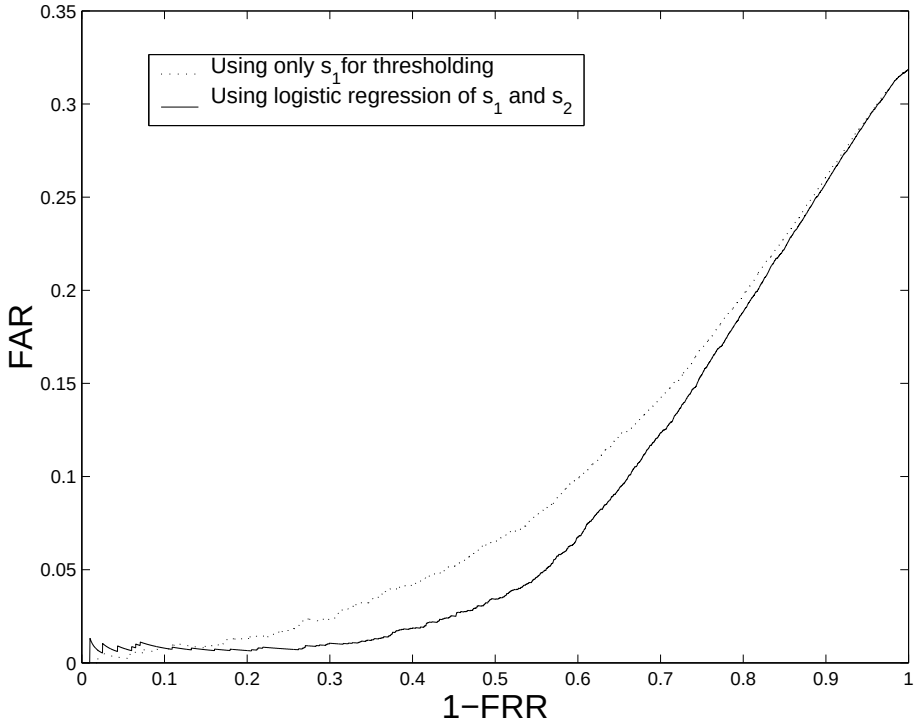


Figure 4.5.1: ROC curves for handwritten word recognition.

4.5.2 Barcode Recognition

In this section we deal with automated recognition of barcodes on mail pieces. We consider 4-state type barcodes with 65 bars. Each bar can be represented by a pair of bits $(b_1, b_2), b_i \in \{0, 1\}$ indicating the presence ($b_i = 1$) or absence ($b_i = 0$) of upper (b_1) or lower (b_2) part of the bar. Thus the whole barcode can be represented by a sequence of 65*2=130 bits $B = (b_1, b_2, \ldots, b_{130})$.

Barcode employs error correction using Reed-Solomon encoding [48] with symbols over Galois field GF(64). It takes 6 bits to encode one symbol, thus a barcode can be represented as $\lceil \frac{130}{6} \rceil = 22$ symbols (one symbol is shorted). Out of these 22 symbols

we have 4 error-correction symbols. The property of the Reed-Solomon encoding is that minimum distance between two codewords is the number of error-correction symbols plus 1. Thus in our case the minimum distance is $4 + 1 = 5$. This means that given one valid barcode we have to change at least 5 symbols (6-bit sequences) to get another valid barcode. Correspondingly, at least 5 bits should be flipped to change one valid barcode into another valid barcode.

The noise model is introduced where it is assumed that any bit can be corrupted and have a new float value in the interval $[0, 1]$. Let us denote the corrupted barcode as a sequence of 130 float numbers $F = (f_1, f_2, \ldots, f_{130})$. The problem of barcode recognition is given a corrupted barcode F we must find some valid barcode B which is closest to F in some sense. Reed-Solomon decoding algorithm operating on binary strings is able to correct 2 corrupted symbols. Decoding of barcode F with float numbers reflecting the probability of upper or lower part of the bar presence involves making hypothesis about proper binary form of the barcode, and combining it with binary Reed-Solomon decoding. In fact, by means of accepting a multitude of hypothesis on what bars are and which bars are corrupted, we are searching through a set of close valid binary barcodes and find the closest one. We do not present noise model and particular distance function $dist(B, F)$ used, since they are irrelevant for current paper.

The performance of the decoding algorithm is measured by the cost function $Cost$ which is a linear combination of the cost of rejecting recognition results, $RejectRate$, and the cost of accepting incorrectly recognized barcode, $MisreadRate$. Three cost functions were considered: $Cost = RejectRate + k * MisreadRate, k = 2, 10, 100$.

To minimize the cost of barcode recognition we need to find the best possible decision algorithm on whether barcode with the best score will be accepted as a recognition

result or not. One of the decisions is based on the comparison of this best score s_1 with some preset threshold. Another decision is based on finding two closest valid barcodes and comparing linear combination of corresponding scores $\alpha_1 s_1 + \alpha_2 s_2$ with preset threshold. The thresholds and parameters α_i were found so that $Cost$ is minimized. Table 4.5.1 presents the results of the experiments. The numbers in the table show minimum values of $Cost$ (expressed in %) given optimal parameters minimizing that cost.

Cost model	Using s_1	Using s_1 and s_2
k=2	0.3261	0.1998
k=10	0.5287	0.2449
k=100	0.9271	0.5194

Table 4.5.1: Costs of barcode recognition are significantly reduced when s_2 is used for thresholding.

Incorporating the score of the second-best matched barcode allows reducing the recognition cost in half. This is a significant improvement given that the recognition algorithm is only slightly modified. Barcode recognition is exactly the identification problem, since we know all the classes - all valid barcodes (2^{106} total), and we are certain during recognition that the corrupted barcode has its truth among all these barcodes. By the linearity property, each valid barcode has a similar neighborhood and impostors are separated from a genuine barcode by some minimum distance. The matching scores s_1 and s_2 are dependent mostly due to the second cause (section 4.3): if corruption is small, then genuine score is high and other scores are low. In case of major corruption, all scores are low. The score dependence seems to have a positive effect on identification model improvement in this situation.

4.5.3 Biometric Person Identification

We consider the problem of person identification by means of fingerprint and face biometrics. We use the NIST Biometric Score Set, release 1 (BSSR1 [1]). We consider three subsets of scores: fingerprint li set which is produced for 6000 enrolled left index fingerprints and 6000 user input fingerprints, face recognizer C set and face recognizer G set which are produced for 3000 enrolled faces and 6000 user input faces. Thus all sets have 6000 identification trials with 1 genuine scores and 5999 (for fingerprints) or 2999 (for faces) impostor match scores.

We ran all experiments using leave-one-out method, that is for each identification trial we use all other 5999 trials for training and perform testing on the left out trial. All 6000 test runs are combined together in the ROC curve. For each test run we reconstruct densities of scores for two classes - genuine identification trials with the best score being the genuine match and impostor identification trials with the best score being impostor match. A test run score is given as a ratio of genuine density to the impostor density at a test trial score point (Bayesian classification).

Results of the experiments are shown in Figures 4.5.2 - 4.5.4. 1 top score thresholding means that we consider only the top scores for reconstructing the densities, and 2 top score thresholding means that we use both the best and second-best scores to reconstruct densities. In all cases making acceptance decision based on two scores has clear advantage over decisions based on just the first score.

The causes for score dependence in biometric person identification are mostly different from those considered in the barcode recognition application. There is no enforced requirement of minimum distance between biometric templates. Thus biometric templates could be close to each other, and even small corruption of any one template

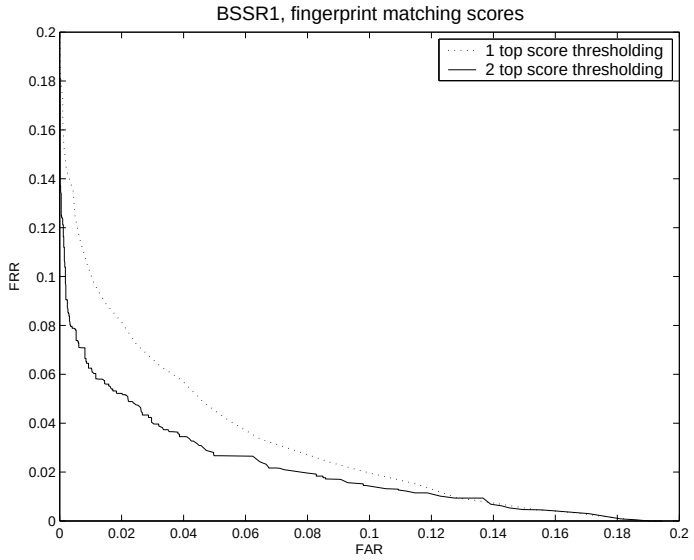


Figure 4.5.2: ROC curves for optimal thresholding using and not using second-best score. BSSR1 set, fingerprint li scores.

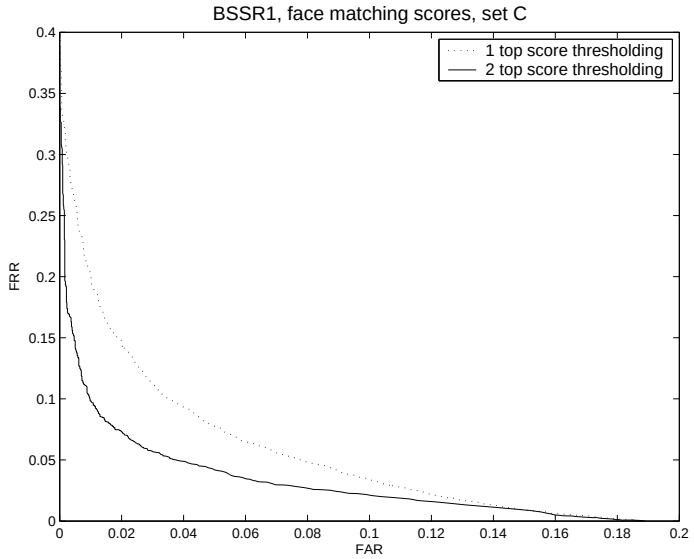


Figure 4.5.3: ROC curves for optimal thresholding using and not using second-best score. BSSR1 set, face recognizer C scores.

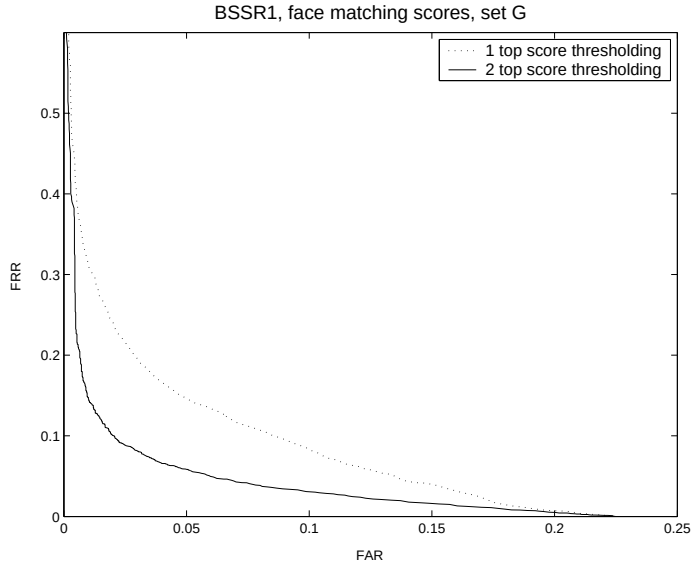


Figure 4.5.4: ROC curves for optimal thresholding using and not using second-best score. BSSR1 set, face recognizer G scores.

could result in impostor match. It seems that the quality of input produces most score dependence in this case. For example, if an input fingerprint has small area, then it will contain small number of minutia, and consequently all minutia based matchers will produce low scores. And if the input fingerprint has many minutia, then not only the genuine match will have many matching minutia and high score, but impostor matches will also get a chance to produce a higher score.

4.6 Conclusion

The purpose of artificial examples relying on matching scores independence assumption explored in sections 4.2 and 4.4 was to reveal that the benefit from using a

combination of scores, instead of only one best score, comes naturally. The improvement is explained by explicitly stating that we deal with the identification process - the true class is one among N matched classes. In case of dependent scores, the total improvement of using score combination can be considered as composite of two parts: improvement due to identification process assumption and improvement due to score dependency. As real life applications show, improvements due to score dependency can be significant.

The score normalization technique is widely used in speaker verification and identification by making implicit assumptions on matching and non-matching score distributions. These assumptions result in implicitly accepting the identification model. Consequently, improvements from using score normalizations techniques can be explained directly by utilizing benefits of the identification process. Part of the improvements can also be explained by utilizing score dependencies.

The questions which is not addressed in this work is whether it helps to use scores $s_3, s_4, \ldots$ in combination. Though the idea of utilizing additional scores is valid and could improve identification, it requires significant increase in the number of training samples.

Chapter 5

Combinations Utilizing Identification Model

In this chapter we consider a combination of matchers in the identification system. A typical application is that of biometric person identification, where M multiple biometric matchers are used to produce MN matching scores, and N is the number of enrolled persons. We assume that M is small and N is large. This problem is the same the classifier combination problem with large number of classes where each classifier outputs N matching scores related to all classes. We use terms 'matcher' to refer to a classifier which outputs class matching scores, and 'identification' to refer to the case dealing with large number of classes.

We investigate combination approaches of the medium II complexity type. As we will show by means of an artificial example, such approaches can result in better performance than traditional low complexity combination approaches. We will also describe methods of combining matching scores to retain information about individual

matchers.

5.1 Previous Work

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 (section 2.2.2). Combination functions can also be user specific - f_i [34, 19] (medium I complexity type). These are in fact the same combination methods which are used in biometric verification problems where person claims his or her identity and the system only matches against enrolled templates of that identity. Thus in verification applications only scores related to one person are available for combination.

However, identification systems produce more scores, namely matching scores for all persons in the database (we assume simple identification systems with no indexing). These additional scores can be used to create an improved combination algorithm. Experiments based on utilizing the second-best score for accepting identification results were presented in the previous chapter. Our results show significant benefits resulting from using both the best and the second-best scores in order to accept or reject a class corresponding to the best score. Is it possible to apply similar techniques to improve the performance of the combination algorithm? This chapter addresses this question and proposes methods of combination involving these additional output scores.

Another type of approach to combination in identification systems is to use rank information of the scores. This approach transforms combination problems with output

type III classifiers to combination problems with output type II classifiers (see section 1.1.3 for the definition of these types). 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 [25]. 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 [11] considered a hybrid approach where traditional combination of matching scores is fused with rank information in order to achieve identification decision. Hong and Jain [29] consider ranks, not for combination, but for modeling or normalizing classifier output score. Saranli and Demirekler [51] 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 from section 2.2.2 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.

Ranking of the matching scores is somewhat similar to the score normalization. Usually score normalization [33] 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.

Score normalization techniques have been well developed in the speaker identification problem. For example, cohort normalizing method [50, 15] 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. are T- and Z- normalization techniques are similar[47]. One of our approaches also relies on a similar likelihood ratio. However, we consider such normalizations in the context of combinations.

5.2 Low Complexity Combinations in Identification System

In this section we show an artificial example of an identification system where low complexity combinations can only provide suboptimal solutions. The example proves

that it is preferable to consider at least medium II complexity combinations for identification systems.

We repeat an example from section 4.3 where we considered different cases of the dependencies in identification model. Suppose we have an identification system with one matcher and, for simplicity, $N = 2$ classes. Also suppose we collected data on the distributions of the genuine and impostor scores and reconstructed the score densities as shown in figure 5.2.1. Is it possible to deduce from these densities the performance of our identification system?

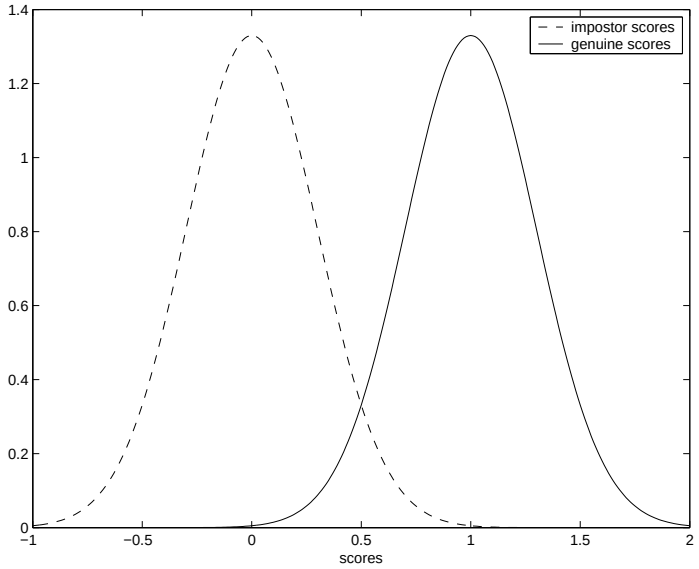


Figure 5.2.1: Hypothetical densities of matching(genuine) and non-matching(impostors) scores.

As in section 4.3, we look at three possible scenarios on how these densities might have originated from the sample of the identification attempts:

1. In every observed identification attempt the score of the genuine class s_{gen} is

chosen from the distribution of genuine scores, and the score of the impostor class s_{imp} is the additive inverse of s_{gen}: $s_{imp} = 1 - s_{gen}$.

2. Both scores s_{gen} and s_{imp} are sampled independently from genuine and impostor distributions.

3. In every observed identification attempt : $s_{imp} = s_{gen} - 1$. Note, this scenario is a little different now, since the best score always corresponds to the genuine class, and previously it was not so.

Let the score index mean the class to which this score was assigned. We previously showed (section 4.3) that in the first scenario, considering both scores s_1 and s_2 for thresholding has no advantages over considering the single best score $\max(s_1, s_2)$. In the second scenario we have some advantage, and in third scenario we have perfect decision. Thus, even though in all three cases the distributions of the individual score densities are the same, the performances of the identification systems based on these three scenarios are quite different. The difference could be discerned only by considering the joint distribution of scores (s_1, s_2) obtained during each identification trial instead of considering them separately.

So, how does this difference in performance translate into the combination of classifiers? Consider a combination of two matchers in our two class identification system: one matcher is from the second scenario, and the other is from the third scenario. 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.

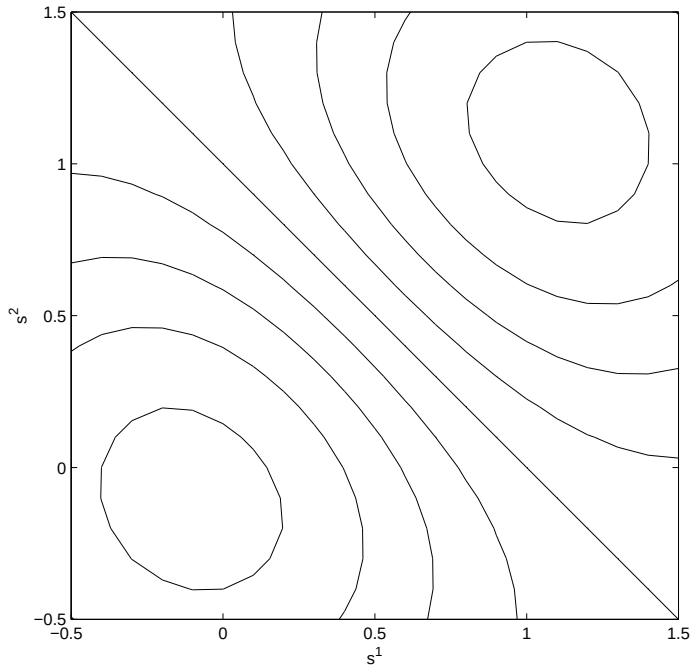


Figure 5.2.2: Optimal decision surfaces for low complexity combinations.

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 5.2.2 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 5.2.2 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.

Note also that medium I complexity type combinations would also discard score dependencies, and a similar example can be constructed to prove that such combinations lead to suboptimal system performance.

5.3 Combinations Using Identification Model

The example from the previous section showed that we really have to consider medium II or high complexity combinations for identification systems. Also in the chapter describing the combination framework, we noted that high complexity combination might not be appropriate for system with large number of classes. Thus medium II complexity type seems to be the best choice of combination type for identification

problems.

As we noted before, rank based combination methods belong to the medium II complexity type. The problem with rank based methods, however, is that the score information is simply discarded. It is easy to construct an example where small difference in scores will result in big difference in ranks and will confuse the combination algorithm. Altincay and Demirekler [4] presented one such example.

Score normalization methods, which utilize the whole set of scores obtained during a current identification trial in order to normalize a score related to a particular class, followed by some combination algorithm, remain a viable approach to combination in identification systems. The question is, of course, what is the proper way to do such normalizations. The same paper by Altincay and Demirekler gives examples where normalization procedures frequently used result in a loss of information contained in a classifier's scores and yield suboptimal classification.

Next, we present different combination methods of medium II complexity type. Score normalization methods developed in the speaker identification research use the term 'background model' to describe the probabilities associated with the event that a considered class is an impostor class during the current identification attempt. Our term 'identification model' has a different meaning and describes the dependencies between scores output for all classes during any one identification attempt.

5.3.1 Combinations by Modeling Score Densities

Suppose that we combine M independent classifiers, and each classifier outputs dependent scores. We need to use normalization by identification model for each output score. The identification system is essentially the classifier combination problem, classifier combination is itself a classifier, and the Bayesian classifier chooses the class which maximizes the posterior class probability. An input is a whole set of scores output by all the M combined classifiers. Thus the goal of classification is to find

$$\arg\max_k P(C_k|\{s_i^j\}_{i=1,\ldots,N;j=1,\ldots,M})$$

Term C_k refers to the fact that the class k is the genuine class. By the Bayes theorem

$$P(C_k|\{s_i^j\}_{i=1,\ldots,N;j=1,\ldots,M}) = \frac{p(\{s_i^j\}_{i=1,\ldots,N;j=1,\ldots,M}|C_k)P(C_k)}{p(\{s_i^j\}_{i=1,\ldots,N;j=1,\ldots,M})}$$

and since the denominator is the same for all classes, our goal is to find

$$\arg\max_k p(\{s_i^j\}_{i=1,\ldots,N;j=1,\ldots,M}|C_k)P(C_k)$$

or, assuming all classes have the same prior probability,

$$\arg\max_k p(\{s_i^j\}_{i=1,\ldots,N;j=1,\ldots,M}|C_k)$$

By our current assumption, classifiers are independent, which means that any subset of scores produced by one classifier is statistically independent from any other subset of scores produced by another classifier. Hence, our problem is to find

$$\arg\max_k \prod_j p(\{s_i^j\}_{i=1,\ldots,N}|C_k) \tag{5.3.1}$$

The problem now is to reliably estimate the densities $p(\{s_i^j\}_{i=1,\ldots,N}|C_k)$, which is a rather hard task given that the number N of classes is large and we do not have many

samples of each class for training. The last problem is solved by noticing that we do not construct class specific combination, and thus class indexes can be permuted. Thus all training samples pertaining to different genuine classes can be used to train only one density, $p(s_k, \{s_i^j\}_{i=1,\ldots,N,i\neq k}|C_k)$. Now s_k^j is a score belonging to genuine match, and all other scores $\{s_i^j\}_{i=2,\ldots,N}$ are from impostor matches. Since there are many impostor scores participating in this density, we might somehow try to eliminate them. Recall, that when considering identification models for decision, we relied on the second best score output by the classifier. Could we use similar consideration and rely only on one or two impostor scores?

Indeed, instead of $p(s_k, \{s_i^j\}_{i=1,\ldots,N,i\neq k}|C_k)$ we can consider $p(s_k^j, t_k^j|C_k)$, where t_k^j is a best impostor score for classifier j, given that the genuine class is k. Note that if s_k is the best matching score, then t_k^j is the second best score, and if s_k is the best score, then t_k^j is the best score. Thus the combination rule is the following:

$$\arg\max_k \prod_j p(s_k^j, t_k^j|C_k) \tag{5.3.2}$$

5.3.2 Combinations by Modeling Posterior Class Probabilities

As above we consider posterior class probability, apply Bayes formula, but now use independence of classifiers to decompose the denominator:

$$\begin{aligned} P(C_k|\{s_i^j\}_{i=1,\ldots,N;j=1,\ldots,M}) &= \frac{p(\{s_i^j\}_{i=1,\ldots,N;j=1,\ldots,M}|C_k)P(C_k)}{p(\{s_i^j\}_{i=1,\ldots,N;j=1,\ldots,M})} \\ = \frac{\prod_j p(\{s_i^j\}_{i=1,\ldots,N}|C_k)P(C_k)}{\prod_j p(\{s_i^j\}_{i=1,\ldots,N})} &= P(C_k)\prod_j \frac{p(\{s_i^j\}_{i=1,\ldots,N}|C_k)}{p(\{s_i^j\}_{i=1,\ldots,N})} \end{aligned} \tag{5.3.3}$$

The next step is similar to the step in deriving the algorithm for background speaker model [47]. We consider class $\overline{C_k}$ meaning some other class is genuine, and decompose

$p(\{s_i^j\}_{i=1,\ldots,N}) = P(C_k)p(\{s_i^j\}_{i=1,\ldots,N}|C_k) + P(\overline{C_k})p(\{s_i^j\}_{i=1,\ldots,N}|\overline{C_k})$. By assuming that N is large and $P(\overline{C_k}) \gg P(C_k)$, we can discard the first term and get the following classifier decision:

$$\arg\max_k \prod_j \frac{p(\{s_i^j\}_{i=1,\ldots,N}|C_k)}{p(\{s_i^j\}_{i=1,\ldots,N}|\overline{C_k})} \tag{5.3.4}$$

In comparison with decision 5.3.1 of the previous section we have have additional density $p(\{s_i^j\}_{i=1,\ldots,N}|\overline{C_k})$. Such density can be viewed as a background of impostors for the genuine class C_k. As research in speaker identification suggests, utilizing such background model is helpful.

One way to model these ratios could be a direct reconstruction of the posterior class probabilities (ratios in equation 5.3.3 are exactly these probabilities without priors). The other way is by additional modeling of $p(\{s_i^j\}_{i=1,\ldots,N}|\overline{C_k})$. We used an approach similar to the previous section to estimate this density as $p(s_k, t_k^j|\overline{C_k})$, but t_k^j now is not the best impostor (we do not know what score is genuine, and thus can not know the best impostor), but simply the second best score.

The technique described in this section can be characterized as a composition of identification model and background model. The identification model considers $p(s_k, t_k^j|C_k)$ and $p(s_k, , t_k^j|\overline{C_k})$ instead of $p(s_k|C_k)$ and $p(s_k|\overline{C_k})$, and background model considers $p(s_k, , t_k^j|\overline{C_k})$ or $p(s_k|\overline{C_k})$ in addition to $p(s_k, t_k^j|C_k)$ or $p(s_k|C_k)$. The background model makes score normalization under the assumption of the independence of scores assigned to different classes, and identification model accounts for dependencies of scores.

5.3.3 Combinations of Dependent Classifiers

The combination algorithms presented in the previous two sections deal with independent classifiers. How should we address dependent classifiers?

By looking at the combination formulas 5.3.1 and 5.3.4 we can see that each classifier contributes terms $p(\{s_i^j\}_{i=1,\ldots,N}|C_k)$ and $\frac{p(\{s_i^j\}_{i=1,\ldots,N}|C_k)}{p(\{s_i^j\}_{i=1,\ldots,N}|\overline{C_k})}$ correspondingly to the combination algorithm. Thus one can conclude that it is possible to model the same terms for each classifier, and then combine them by some other trainable function.

Note that any relationships between scores $s_{i_1}^{j_1}$ and $s_{i_2}^{j_2}$ where $i_1 \neq i_2$ and $j_1 \neq j_2$ will be essentially discarded. This seems to be inevitable for the current complexity type of combinations - medium II. If we wanted to account for such relationships, we would need class-specific combination functions, and thus higher complexity combinations.

5.3.4 Normalizations Followed by Combinations and Single Step Combinations

Figure 5.3.1 represents in graphical form the type of combinations we have presented thus far. All these combinations consist of two steps. In the first step, each score is normalized by using other scores output by the same matcher. In the second step, normalized scores are combined using a predetermined or trained combination function.

However, it is not necessary to have these two steps for combinations. For each normalization happening in the first step we use the same identification model statistic and the same trained density estimates. Thus the contribution of the particular

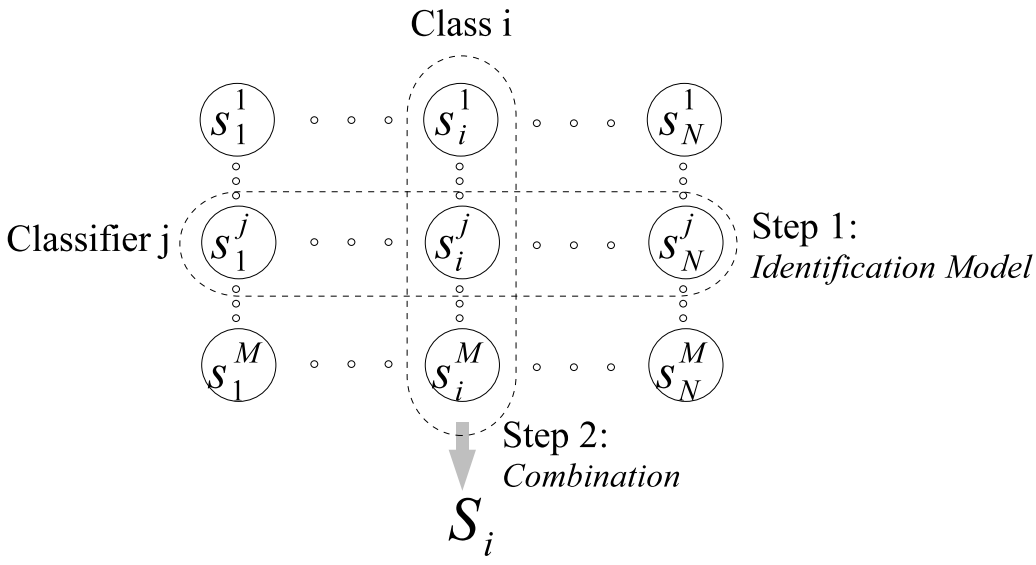


Figure 5.3.1: 2-step combination method utilizing identification model.

classifier j to the whole combination algorithm's output for class i is calculated only from score s_i^j and statistic t^j (statistic though could vary for a class; in first case it was best or second best score; thus in fact two values are used). Figure 5.3.2 presents a diagram on how scores and statistics from all participating classifiers could be combined in a single combination step.

In the algorithm presented by this diagram the statistics t^j are extracted for each classifier j using its output scores by a predetermined and non-trainable algorithm. The scores related to a particular class and statistics are combined together by a trainable function. This combination function is not class-specific and is easily trainable. This type of combinations are of medium II complexity type. In comparison, in low complexity type combinations only scores for a particular class are combined, and not statistics from identification models of classifiers.

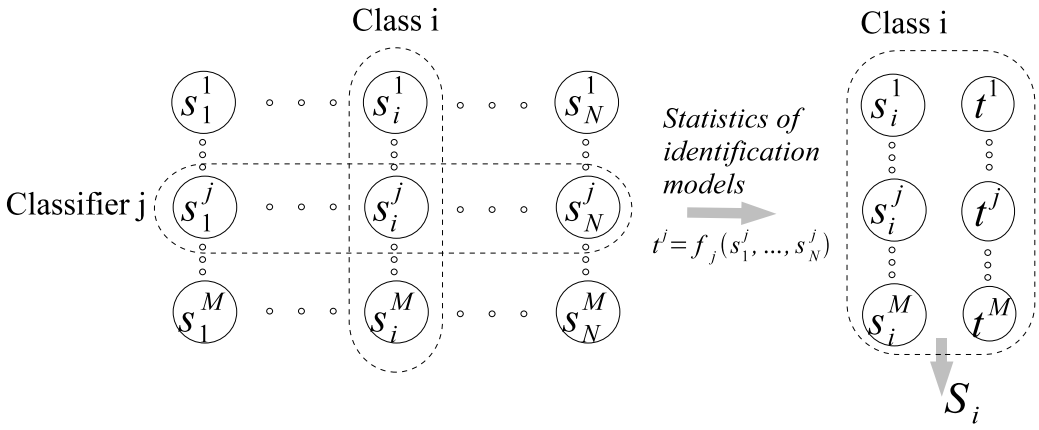


Figure 5.3.2: 1-step combination method utilizing identification model.

5.4 Experiments

We conducted experiments using NIST BSSR1 biometric score database [1]. We used subsets of left fingerprint (li) and two subsets of face scores from two face matchers C and G. Since we wanted to consider first the case of independent matchers we performed two sets of experiments on combining fingerprint and face scores : li combined with C, and li combined with G.

Results are presented in Table 5.4.1. The columns represent the combination method. 'Low Complexity' 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. 'Density' is the method outlined in section 5.3.1. 'Posterior Probability' is the method from section 5.3.2. 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.

All experiments were performed in leave-one-out framework. The numbers in the

Matchers	Number of tests	Low Complexity	Density	Posterior Probability
li & C	516	5	7	4
li & G	517	9	11	6

Table 5.4.1: Experiments on combinations in identification systems. Entries are the numbers of failed test identification trials.

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 algorithm for low complexity combinations is exactly the same as was presented in section 3.5. For each pair of scores the combined score is derived as a ratio of genuine and impostor density function approximations at this score pair. Thus, this combination method automatically deals with the background model - the density of impostors participates in the combined score. This might explain why low complexity combinations got better results than combinations based on genuine score density approximation as section 5.3.1 ('Density' method in table 5.4.1). But if identification model is combined with background model as in section 5.3.2 ('Posterior Probability' method), then we are able to obtain better combination than the low complexity method.

5.5 Identification Model for Verification Systems

Although there are examples where score normalization techniques with background models are used for speaker identification tasks[11], even more applications use such

techniques for speaker verification systems [50, 54, 47]. We also applied the combinations utilizing identification models for biometric person verification tasks. The drawback of using either the background models or the identification models in verification tasks is that we have to produce not only one match per person and per matcher, but also some set of matching scores for other persons enrolled in the system, or some artificially modeled persons.

In our experiments for each test person we performed match of input biometric with biometric templates of all enrolled persons. All these scores were used to derive a score normalized by identification and background models as in section 5.3.2 ('Posterior Probability' method). The ROC curves were obtined by means of thresholding these normalized scores for both genuine and impostor verification attempts. These curves are drawn in Figures 5.5.1 and 5.5.2 together with corresponding ROC curves taken from section 3.5 corresponding to traditional low complexity combinations.

We distinguish two possible cases with respect to impostors in such verification systems: impostor is enrolled in the database, and impostor is not enrolled in the database. If the impostor is in the database, and impostor attempts to be verified as another person, we expect that match score to the true impostor's identity will be higher than impostor's match score to the claimed identity. Thus a verification system utilizing the identification model (and hence all matching scores) is more likely to reject this impostor's matching attempt. Experimental results in Figures 5.5.1 and 5.5.2 show that the performance of verification system is better if impostor is enrolled in the database than when the impostor is not in the database. But this difference in performance is small, and both cases have better performance than traditional combination of low complexity type. The small difference in performance can be explained by the fact that our identification model algorithm uses second-best impostor statistics instead of best impostor statistics (section 5.3.2).

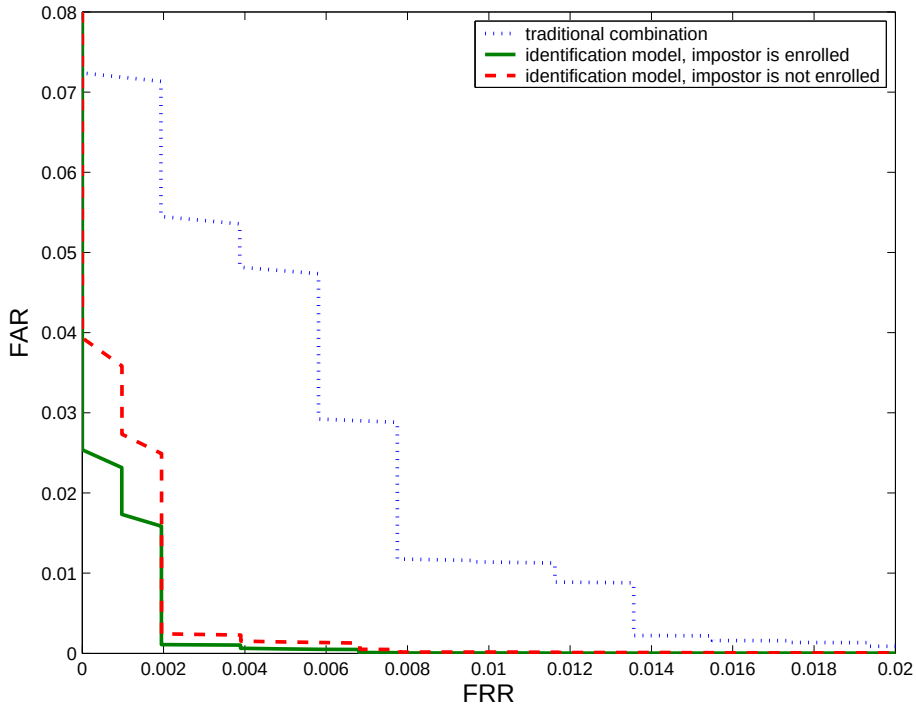


Figure 5.5.1: ROC curves for traditional low complexity combinations and combinations utilizing identification models in verification tasks. BSSR1 set, fingerprint li and face C scores.

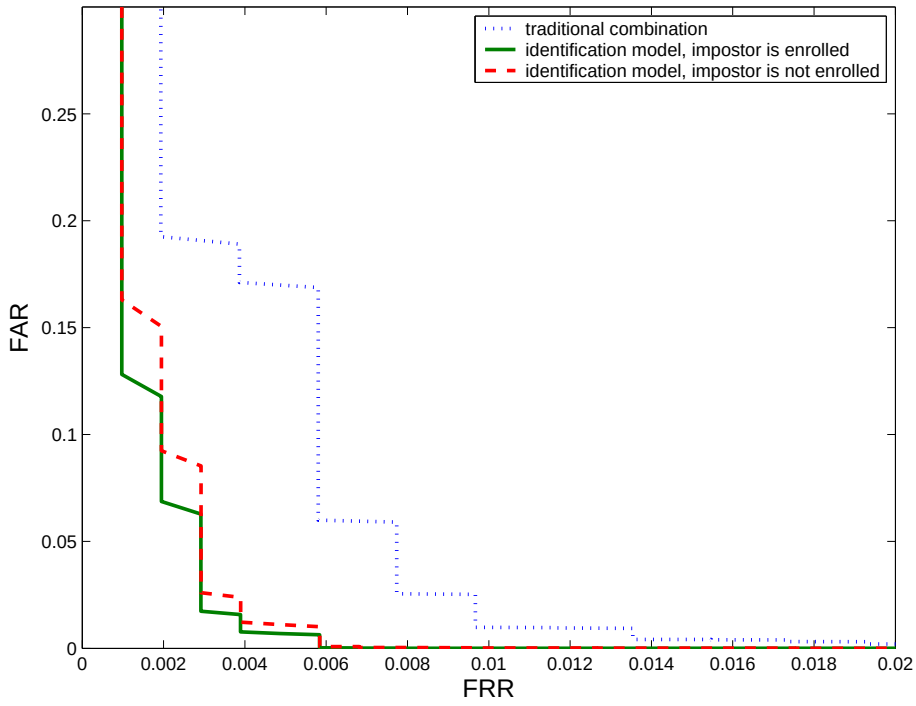


Figure 5.5.2: ROC curves for traditional low complexity combinations and combinations utilizing identification models in verification tasks. BSSR1 set, fingerprint li and face G scores.

Figures 5.5.1 and 5.5.2 show that we were able to achieve significant improvement in the performance of verification systems. These improvements seem to be similar to the improvements achieved by using identification models for making acceptance decisions in biometric person identification in section 4.5.3.

5.6 Conclusion

In this chapter we presented an example of the combination problem which is optimally solved by the combinations of medium II complexity type, and is not solved adequately by the combinations of low or medium I complexity types. Our example underscores the necessity for considering this type of combinations in identification systems. The example also shows that the addition of another classifier into combination can worsen the performance (even if an optimal combination algorithm is used) when the combination algorithm belongs to non-suitable complexity type.

The main contribution of this chapter is a presentation of combination methods of medium II complexity type utilizing the identification models of involved classifiers. The experiments performed with the sets of biometric matching scores confirm the advantage of the proposed combination methods over low complexity combination methods.

Considered biometric score sets (BSSR1 set, fingerprint and face C and G recognizers) produce slightly dependent scores during each identification attempt (not the scores from different recognizers, but scores from same recognizer). Observed correlation between genuine and best impostor scores for these recognizers is about $0.10-0.15$. It is still not clear whether the improvements mostly come naturally from identification model (as in artificial example with independent scores considered in section 4.2) or

from score dependencies which are accounted for by identification models.

Chapter 6

Conclusions

The contributions of this thesis are:

- Categorization of 4 combinations types based on the complexity of combining functions and the development of a combination framework using these types.
- Investigation of the benefits of using external information about classifiers in the combination process, in particular, knowledge of the independence of classifiers.
- Development of an identification model which can be used for decision making and combinations in identification systems.

Four complexity combination types described in chapter 2 characterize classifier combinations. These types originate naturally from the structure of the combination parameters - each score corresponds to some class and classifier, and the output of the combinator corresponds to some class. If each classifier in the combination produces N confidence scores for $N \geq 2$ classes, then the combination is one of the

four type. In case of 2 classes and classifiers outputting only one matching score, for example, where score 0 corresponds to one class and score 1 corresponds to another class, the score matrix of Figure 2.2.2 will only have one column, and all combination types degenerate to a single low complexity combination.

The combination framework prompts the user to choose the combination complexity type first based on the numbers of classes and classifiers, and the number of training samples. Within a chosen complexity type one can use any generic classifier for combination. Finally, a generic classifier used for combination can be modified to account for a chosen complexity type and for any extraneous information about classifiers.

By viewing existing combination approaches from the positions of developed framework, it might be possible to predict if the used combination algorithm can be improved in some way. For example, if we know that classifiers are independent, as in the case of multimodal biometric matchers, the combination algorithm can utilize this information. In chapter 3 we investigated the performance of Bayesian classifiers utilizing and not utilizing this information, and we saw both experimentally and theoretically that such utilization can be quite beneficial. Clearly, if some existing combination algorithm, say SVM or neural network, is not utilizing such information, we can state that such algorithm can be somehow modified to produce better combination results.

As another application of our combination framework, we observe that frequently the algorithms, employed for combining matchers in biometric identification systems, only use the scores related to one class to produce a final combination score. Such algorithms simply discard the dependency information between scores assigned to all classes by one classifier. We gave an example, 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 [49] 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 example (section 5.2) low complexity combinations possess the isolation property, but 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.

In order to reflect the relationships between scores assigned by one classifier to different classes, we introduced the concept of the identification model. The identification model application is a score normalization algorithm where normalization depends on all scores output by a classifier in one identification trial, and the algorithm is the same for all classes. Thus our identification model has less complexity than similar attempts to normalization [9, 31]. In these previous attempts normalizations were class specific and required huge amount of training data. The combinations utilizing such normalizations will be similar to behavior-knowledge space combination [44], and they belong to high complexity combination type. Biometric identification problems can have large number of enrolled persons, and such combinations are not feasible due to the lack of training data. By restricting ourselves to non-class-specific normalizations of the identification model we are able to concentrate on combinations of medium II complexity type. Such combinations have significantly lower complexity,

and result in efficient algorithms for identification systems.

In chapter 4 we have showed how the identification model can be used in order to improve the performance of decision making in identification systems, and chapter 5 contains examples of combination algorithms utilizing identification models of each involved classifier. The experiments show significant benefits in using identification models, as opposed to less efficient low complexity type combinations, and non-feasible high complexity combinations.

Appendix A

Complexity of Functions with N-dimensional Outputs

Consider the definition of VC dimension using concept of function set shattering a set of points. Using notation of [60] consider a set of points $z_1, \ldots, z_l$ and a set of indicator functions $Q(z, \alpha), \alpha \in \Lambda$. Let $N^{\Lambda}(z_1, \ldots, z_l)$ denote the number of different separations of set $z_1, \ldots, z_l$ by functions $Q(z, \alpha), \alpha \in \Lambda$. The main theorem related to the definition of VC dimension states that either $\sup_{z_1,\ldots,z_l} N^{\Lambda}(z_1, \ldots, z_l) = 2^l$ for any l, or there is some integer h such that $\sup_{z_1,\ldots,z_l} N^{\Lambda}(z_1, \ldots, z_l) = 2^l$ for $l \leq h$ and $\sup_{z_1,\ldots,z_l} N^{\Lambda}(z_1, \ldots, z_l) \leq \sum_{i=0}^{h} C_l^i \leq \left(\frac{el}{h}\right)^h$ for $l > h$. The VC dimension of a function set $Q(z, \alpha), \alpha \in \Lambda$ equals infinity in the first case, and equals h in the second case.

For simplicity consider a case of two classes in the classifier combination framework (N=2) first. Thus there are two output scores S_1 and S_2 where S_i was calculated by some function from the set $Q(z, \alpha_i), \alpha_i \in \Lambda_i, i = 1, 2$. Generally, function sets

$Q(z, \alpha_i), \alpha_i \in \Lambda_i$ are different for different i and training (search for functions calculating S_1 and S_2) can be performed separately.

If we have a set of points $z_1, \ldots, z_l$ and two indicator functions $Q(z, \alpha_i), \alpha_i \in \Lambda_i, i = 1, 2$ defined on this set, then all possible values of these functions are $(0, 0), (0, 1), (1, 0)$ and $(1, 1)$. So the set of points $z_1, \ldots, z_l$ is separated into 4 subsets corresponding values of two indicator functions. In general, for functions having outputs in N-dimensional space, the point set will be split into 2^N subsets.

Definition A.1 *Define $N^{\Lambda_1,\ldots,\Lambda_N}(z_1, \ldots, z_l)$ as the number of different separations of a point set $z_1, \ldots, z_l$ into 2^N subsets by all possible combinations of indicator functions $Q(z, \alpha_i), \alpha_i \in \Lambda_i, i = 1, \ldots, N$.*

If functions $Q(z, \alpha_1), \alpha_1 \in \Lambda_1$ and $Q(z, \alpha_2), \alpha_2 \in \Lambda_2$ separate set $Z = \{z_1, \ldots, z_l\}$ into corresponding subsets $Z_1^1 \bigcup Z_1^2 = Z$ and $Z_2^1 \bigcup Z_2^2 = Z$ then together they separate Z into subsets $Z_1^1 \bigcap Z_2^1$, $Z_1^1 \bigcap Z_2^2$, $Z_1^2 \bigcap Z_2^1$ and $Z_1^2 \bigcap Z_2^2$. Reverse is also true: any separation of Z into four subsets by functions $Q(z, \alpha_i), \alpha_i \in \Lambda_i, i = 1, 2$ corresponds to two separations of set Z by functions $Q(z, \alpha_1), \alpha_1 \in \Lambda_1$ and $Q(z, \alpha_2), \alpha_2 \in \Lambda_2$ individually. Thus it is easy to see that $N^{\Lambda_1,\Lambda_2}(z_1, \ldots, z_l) \leq N^{\Lambda_1}(z_1, \ldots, z_l) N^{\Lambda_2}(z_1, \ldots, z_l)$. In general, for N-dimensional case we will have

$$N^{\Lambda_1,\ldots,\Lambda_N}(z_1, \ldots, z_l) \leq \prod_{i=1}^{N} N^{\Lambda_i}(z_1, \ldots, z_l) \tag{A.0.1}$$

Since VC dimension characterizes the growth of function $N^{\Lambda}(z_1, \ldots, z_l)$, let us investigate the growth of function $N^{\Lambda_1,\ldots,\Lambda_N}(z_1, \ldots, z_l)$. Maximum number of all separations of l points into 2^N subsets is $(2^N)^l$. Thus $N^{\Lambda_1,\ldots,\Lambda_N}(z_1, \ldots, z_l) \leq (2^N)^l$. If for all $i, i = 1, \ldots, N$ function sets $Q(z, \alpha_i), \alpha_i \in \Lambda_i$ completely separate points $z_1, \ldots, z_l$, that is $N^{\Lambda_i}(z_1, \ldots, z_l) = 2^l$, then it is possible that $N^{\Lambda_1,\ldots,\Lambda_N}(z_1, \ldots, z_l) = (2^N)^l$. Note

that such equality is possible if functions $Q(z, \alpha_i)$ can be chosen independently for different dimensions i. If functions $Q(z, \alpha_i)$ are dependent on each other in combination, the growth of function $N^{\Lambda_1,\ldots,\Lambda_N}(z_1, \ldots, z_l)$ can be significantly smaller. For example, for 2-dimensional case, if $Q(z, \alpha_1) = 1 - Q(z, \alpha_2)$, $N^{\Lambda_1,\Lambda_2}(z_1, \ldots, z_l) = N^{\Lambda_1}(z_1, \ldots, z_l)$. Further properties of the growth of function $N^{\Lambda_1,\ldots,\Lambda_N}(z_1, \ldots, z_l)$ are summarized in the following theorem.

Theorem A.1 *If any of the function sets $Q(z, \alpha_i), \alpha_i \in \Lambda_i, i = 1, \ldots, N$ has infinite VC dimension then*

$$\ln \sup_{z_1,\ldots,z_l} N^{\Lambda_1,\ldots,\Lambda_N}(z_1, \ldots, z_l) = \theta(l)$$

If all function sets $Q(z, \alpha_i), \alpha_i \in \Lambda_i, i = 1, \ldots, N$ have finite VC dimensions $h_1, \ldots, h_N$ then for $l > \max_i h_i$

$$\ln \sup_{z_1,\ldots,z_l} N^{\Lambda_1,\ldots,\Lambda_N}(z_1, \ldots, z_l) \le \ln \left(\sum_{i=0}^{H} C_{Nl}^{i} \right) \le H \left(1 + \ln N + \ln \frac{l}{H} \right) \qquad \text{(A.0.2)}$$

where $H = h_1 + \cdots + h_N$.

Proof: If any of the function sets $Q(z, \alpha_i), \alpha_i \in \Lambda_i, i = 1, \ldots, N$ has infinite VC dimension, for example $Q(z, \alpha_1), \alpha_1 \in \Lambda_1$, then for any l there is a set of points $z_1, \ldots, z_l$ such that $Q(z, \alpha_1), \alpha_1 \in \Lambda_1$ separates it into all 2^l combinations of 2 subsets. Then for this set of points $N^{\Lambda_1,\ldots,\Lambda_N}(z_1, \ldots, z_l) \ge N^{\Lambda_1}(z_1, \ldots, z_l) = 2^l$. Thus $2^l \le \sup_{z_1,\ldots,z_l} N^{\Lambda_1,\ldots,\Lambda_N}(z_1, \ldots, z_l) \le (2^N)^l$ and $\ln \sup_{z_1,\ldots,z_l} N^{\Lambda_1,\ldots,\Lambda_N}(z_1, \ldots, z_l) = \theta(l)$.

If all function sets $Q(z, \alpha_i), \alpha_i \in \Lambda_i, i = 1, \ldots, N$ have finite VC dimensions $h_1, \ldots, h_N$ then for $l > \max_i h_i$ due to A.0.1 and properties of VC dimension for one-dimensional functions:

$$N^{\Lambda_1,\ldots,\Lambda_N}(z_1, \ldots, z_l) \le \prod_{i=1}^{N} \left(\sum_{j_i=0}^{h_i} C_l^{j_i} \right)$$

After expansion and regrouping binomial terms and applying Vandermonde's convolution formula we get:

$$\prod_{i=1}^{N}\left(\sum_{j_i=0}^{h_i} C_l^{j_i}\right) \leq \sum_{k=0}^{H}\left(\sum_{j_1+\cdots+j_N=k} C_l^{j_1}\ldots C_l^{j_N}\right) = \sum_{k=0}^{H} C_{Nl}^{k}$$

The rest of inequality A.0.2 follows from the corresponding part of the proof for one-dimensional VC dimension[60]. ■

Based on this theorem we can introduce a definition of VC dimension for indicator functions with N-dimensional outputs.

Definition A.2 *If for a set of indicator functions $Q(z,\alpha_i),\alpha_i \in \Lambda_i, i = 1,\ldots,N$ the number of different separations of a point set $z_1,\ldots,z_l$ into 2^N subsets is more or equal to 2^l for any l ($N^{\Lambda_1,\ldots,\Lambda_N}(z_1,\ldots,z_l) \geq 2^l$) , then this set of functions has infinite VC dimension. Otherwise VC dimension of the set of functions is a smallest H which makes inequality A.0.2 true for all l bigger than maximum of the VC dimensions of individual function sets $Q(z,\alpha_i),\alpha_i \in \Lambda_i$.*

Bibliography

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

[2] L. A. Alexandre, A. C. Campilho, and M. Kamel. On combining classifiers using sum and product rules. *Pattern Recognition Letters*, 22(12):1283–1289, 2001.

[3] F. M. Alkoot and J. Kittler. Experimental evaluation of expert fusion strategies. *Pattern Recognition Letters*, 20(11-13):1361–1369, 1999.

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

[5] P. A. Andersen. *Handbook of communication and emotion.* Academic Press, 1998.

[6] C. C. Beardah and M. Baxter. The archaeological use of kernel density estimates. *Internet Archaeology*, (1), 1996.

[7] S. Ben-Yacoub, Y. Abdeljaoued, and E. Mayoraz. Fusion of face and speech data for person identity verification. *Neural Networks, IEEE Transactions on*, 10(5):1065–1074, 1999.

[8] R. Bolle, J. Connell, S. Panakanti, N. Ratha, and A. Senior. The relation between the roc curve and the cmc. In *Auto-ID*, 2005.

[9] D. Bouchaffra, V. Govindaraju, and S. Srihari. A methodology for mapping scores to probabilities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9), September 1999.

[10] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

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

[12] R. Caruana, A. Niculescu, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *The International Conference on Machine Learning (ICML'04)*, 2004.

[13] S. B. Cho and J. H. Kim. "Combining multiple neural networks by fuzzy integral for robust classification". *IEEE Transactions on Systems, Man, and Cybernetics*, 25(2):380–384, 1995.

[14] R. Clemen and R. Winkler. Combining probability distributions from experts in risk analysis. *Risk Analysis*, 19:187–203, 1999.

[15] 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.

[16] R. M. Cooke. *Experts in Uncertainty: Opinion and Subjective Probability in Science*. Oxford University Press, 1991.

[17] R. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. d. Ridder, and D. Tax. Prtools4, a matlab toolbox for pattern recognition, 2004.

[18] R. P. W. Duin and D. M. J. Tax. Classifier conditional posterior probabilities. *Lecture Notes in Computer Science*, 1451:611–619, 1998.

[19] 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.

[20] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[21] G. Fumera and F. Roli. Performance analysis and comparison of linear combiners for classifier fusion. In *SSPR/SPR*, pages 424–432, 2002.

[22] G. Fumera and F. Roli. Analysis of error-reject trade-off in linearly combined multiple classifiers. *Pattern Recognition*, 37(6):1245–1265, 2004.

[23] P. Grother and P. Phillips. Models of large population recognition performance. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–68–II–75 Vol.2, 2004.

[24] W. Hardle. *Smoothing Techniques with Implementation in S.* Springer-Verlag, 1990.

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

[26] T. K. Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832–844, 1998.

[27] T. K. Ho. Multiple classifier combination: Lessons and next steps. In A. Kandel and H. Bunke, editors, *Hybrid Methods in Pattern Recognition*, pages 171–198. World Scientific, 2002.

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

[29] 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.

[30] Y. S. Huang and C. Y. Suen. "A method of combining multiple experts for the recognition of unconstrained handwritten numerals". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):90–94, 1995.

[31] K. Ianakiev. *Organizing Multiple Experts for Efficient Pattern Recognition*. Ph.D Thesis, SUNY at Buffalo, 2000.

[32] A. Jain, L. Hong, and Y. Kulkarni. A multimodal biometric system using fingerprint, face and speech. In *AVBPA*, 1999.

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

[34] 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.

[35] J.-H. Kim, G.-J. Jang, S.-J. Yun, and Y. H. Oh. Candidate selection based on significance testing and its use in normalisation and scoring. In *5th International Conference on Spoken Language Processing (ICSLP-1998)*, 1998.

[36] F. Kimura and M. Shridhar. "Handwritten numerical recognition based on multiple algorithms". *Pattern Recognition*, 24(10):969–983, 1991.

[37] J. Kittler, M. Hatef, R. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 226–239, March 1998.

[38] E. M. Kleinberg. Stochastic discrimination. *Annals of Mathematics and Artificial Intelligence*, 1, 1990.

[39] E. M. Kleinberg. On the algorithmic implementation of stochastic discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):473–490, 2000.

[40] E. Kong and T. Dietterich. Error-correcting output coding corrects bias and variance. In *12th International Conference on Machine Learning*, pages 313–321, 1995.

[41] L. Kuncheva. A theoretical study on six classifier fusion strategies. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):281–286, 2002.

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

[43] 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.

[44] L. Lam and C. Y. Suen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16(9):945–954, 1995.

[45] M. Last, H. Bunke, and A. Kandel. A feature-based serial approach to classifier combination. *Pattern Analysis and Applications*, 5(4):385–398, 2002.

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

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

[48] W. Peterson and E. Weldon. *Error-Correcting Codes*. MIT Press, Cambridge, USA, 2nd edition, 1972.

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

[50] 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.

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

[52] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[53] R. E. 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.

[54] A. Schlapbach and H. Bunke. Using hmm based recognizers for writer identification and verification. In *9th Intl Workshop on Frontiers in Handwriting Recognition (IWFHR-9 2004)*, 2004.

[55] B. W. Silverman. *Density estimation for statistics and data analysis*. Chapman and Hall, London, 1986.

[56] P. Slavik and V. Govindaraju. An overview of run-length encoding of handwritten word images. Technical Report 2000-09, State University of New York at Buffalo, August 2000.

[57] R. Tibshirani. Bias, variance and prediction error for classification rules. Technical Report 41, Department of Statistics, University of Toronto, 1996.

[58] K. Tumer and J. Ghosh. Linear and order statistics combiners for pattern classification. In A. J. Sharkey, editor, *Combining Artificial Neural Nets: Ensembles and Multi-Net Systems*, pages 127–162. Springer-Verlag, London, 1999.

[59] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.

[60] V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[61] J. Wayman. Error rate equations for the general biometric system. *Robotics & Automation Magazine, IEEE*, 6(1):35–48, 1999.

[62] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–260, 1992.

[63] 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.