Using independence assumption to improve multimodal biometric fusion

Sergey Tulyakov and Venu Govindaraju

SUNY at Buffalo, Buffalo NY 14228, USA

Abstract. There is an increased interest in the combination of biometric matchers for person verification. Matchers of different modalities can be considered as independent 2-class classifiers. This work tries to answer the question of whether assumption of the classifier independence could be used to improve the combination method. The combination added error was introduced and used to evaluate performance of various combination methods. The results show that using independence assumption for score density estimation indeed improves combination performance. At the same time it is likely that a generic classifier like SVM will still perform better. The magnitudes of experimentally evaluated combination added errors are relatively small, which means that choice of the combination method is not really important.

1 Introduction

Use of multiple biometric identification devices requires robust combination algorithms to increase their effectiveness. Though a plethora of possible combination algorithms are described in scientific literature, there seems to be no consensus on which is the best. Hence a practitioner is forced to try different combination strategies in order to see which algorithm works best for the task at hand.

In this paper we shall deal with the combination of independent classifiers. We assume that classifiers output a set of scores reflecting the confidences of input belonging to corresponding class. It is safe to assume that matching scores for biometrics of different modalities (e.g. fingerprint and face) are independent random variables. The matchers or classifiers possessing this property are independent classifiers. This assumption is fairly restrictive for the field of classifier combination since combined classifiers usually operate on the same input. Though frequently using completely different features for different classifiers still results in dependent scores. For example, features can be dependent, image quality characteristic can influence scores of combined classifiers, and input may have an inherent low match to stored class templates giving low scores for all matchers. In certain situation 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). In the case of multimodal biometrics the inputs to different sensors are independent (for example, no connection of fingerprint features to face features

have been noticed so far). Hence matching scores will also be independent. We will use this property in our work.

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. The main purpose of this work is to see if independence assumption can be effectively used to improve the combination results. We will use the property that the joint density of the classifiers' scores is the product of the densities of the scores of individual classifiers.

We choose the magnitude of added error as a measure of combination goodness. It is the difference between an error of optimal Bayesian combination and 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 could be used to estimate added error in real-world classifier combinations. The main purpose of using this technique is to provide some confidence in combination results. This would enable us to say: "'The total combination error in this particular combination is 5% and added error due to combination algorithm is likely to be less than 1%"'.

2 Previous Work

The added error introduced by Tumer and Ghosh[1] was under much consideration lately [2–5]. The definition of this added error requires assumption that combined classifiers operate in the same feature space and 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 added error is the difference of the combination error of trained classifiers and Bayes error.

This framework does not work 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 minimum Bayes cost, of the combination is determined by the score distributions. And we define combination added error, or combination added cost, as a difference between total error(cost) and this Bayes error(cost). The difference with the previous definition is that we use distributions of scores in score space and not distributions of feature vectors in feature vector space for definition of Bayes error. Thus our combination added error in contrast to previously defined added error[1] will not depend on the added errors of individual classifiers but will depend only on the combination algorithm. See section 3.2 for formal definition of combination added error.

To further explain the difference between two types of added error, let us have an example of few imperfect classifiers operating in the same feature space. Suppose that we have optimal combination based on Bayesian classifier operating on scores in score space (assuming the score distribution are known). In this case added error in Tumer and Ghosh's framework will be some non-zero number

reflecting the errors made by classifiers. In our setting the added error is 0, since the combination algorithm is perfect and did not add any errors to the classification results.

Another assumption made in Tumer and Ghosh's added error framework is that the outputs of the classifiers approximate posterior class probabilities $s_i = P(c_i|x)$. Since in that framework all classifiers use same feature vector x, this implies that output scores of classifiers are very strongly correlated. If we think about score space $s_1, \ldots, s_n$, the outputs of these classifiers would be near the diagonal $s_1 = \cdots = s_n$. Decision boundary of combination algorithm can be represented as a hypersurface in score space and combination decision is roughly equivalent to hypersurface intersecting diagonal at same place. So this added error is mostly concerned with what happens locally near diagonal of the score space and how combination algorithm hypersurfaces intersect it. In this situation, any fixed combination rule will produce approximately the same total recognition error, since their hypersurfaces will be able to intersect the diagonal in some optimal place.

In our case we consider a more general case of output scores not approximating posterior class probabilities and present anywhere in the score space. This is a frequent situation with biometric scores which usually represent some internal matching distance. In this situation, the total error would greatly depend on the used combination rule and score densities. So any combination method which has limited number of trainable parameters will be considered suboptimal in this situation. For example, weighted sum rule will have hyperplanes as decision boundaries and would fail to properly separate classes with normally distributed scores and with hyperquadric decision surface.

It would make little sense to define combination added error for such methods and perform its analysis. Indeed, if optimal decision surface is of the form supported by such combination, the added error will be very close to 0 (limited number of parameters, large number of training samples), and its comparison to any other method will be unfair. On the other hand, if optimal decision surface is different from the ones supported by combination algorithm, there will always be some fixed minimum added error, which this combination method would not be able to improve no matter how many training samples we have.

For these reasons we consider only combination algorithms able to approximate any decision functions given a sufficient number of training samples. In particular, we consider combination methods based on non-parametric density estimation, neural networks and support vector machines. We are interested in seeing what magnitude of combination added error they make with respect to different parameters of combination: number of training samples, total error, number of combined classifiers.

3 Combination Methods

In this work we shall deal with 2-class classifiers. As we mentioned earlier, the main motivation for this work is the combination of biometric matchers. The

combination problem in this area is usually split into two tasks: verification and identification. The verification problem asks if a person's identity coincides with the claimed one, and the identification problem asks to which person among k enrolled persons the given biometric readings belong. We investigate the verification problem and assume that there are two classes of events: the claimed identity coincides with the person's identity and claimed identity is different from the person's identity.

Though two classes are considered, only one score for each matcher is usually available - matching score between input biometric and 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 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 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 ith place. In this case the scores for n classifiers will be located in nk-dimensional space and the classification problem will be more difficult. This framework will be suitable for a biometric identification problem, and should be addressed in the future work.

Since we assume that we combine independent classifiers, is it possible to use this information to design better combination than generic 2-class n-dimensional classifier? The idea is that it might be possible to better estimate joint score density of n classifiers as a product of n separately estimated score densities of each classifier. Effectively, an n-dimensional (for 2 classes) combination problem will be reduced to n 1-dimensional density estimation problems. The question is will this combination based on density products perform better than generic pattern classifiers?

3.1 Combination Using Products of Density Functions

Consider a combination problem with n independent 2-class classifiers. Denote the density function of scores produced by 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. Denote the region of n-dimensional score space being classified by 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 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

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

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

3.2 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 1 with these approximations will result in decision regions R'_i which ordinarily will not coincide with optimal Bayesian decision regions R_i. The combination added cost(AC) will be defined as a difference between cost of using trained regions R'_i and optimal regions R_i for combination:

$$AC = \lambda_1 P_1 \int_{R'_2} p_1(\mathbf{x})d\mathbf{x} + \lambda_2 P_2 \int_{R'_1} p_2(\mathbf{x})d\mathbf{x}$$
$$- \lambda_1 P_1 \int_{R_2} p_1(\mathbf{x})d\mathbf{x} - \lambda_2 P_2 \int_{R_1} p_2(\mathbf{x})d\mathbf{x} \quad (2)$$

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

$$AC = \int_{R'_2 \cap R_1} \left(\lambda_1 P_1 p_1(\mathbf{x}) - \lambda_2 P_2 p_2(\mathbf{x})\right) d\mathbf{x}$$
$$+ \int_{R'_1 \cap R_2} \left(\lambda_2 P_2 p_2(\mathbf{x}) - \lambda_1 P_1 p_1(\mathbf{x})\right) d\mathbf{x} \quad (3)$$

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

4 Experiments

The experiments were performed for two normally distributed classes with means at (0,0) and (1,1) and different variance values (same for both classes). It was also assumed that costs and prior probabilities of both classes are equal. The Bayesian decision boundary in this situation is a straight line $x + y = 1$. Note that both 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 previous section is denoted here as DP. The kernel density estimation method with normal kernel densities [6] was used for estimating one-dimensional score densities. We chose least-square cross-validation method for finding a smoothing parameter. Arguably, the choice of normal kernel would favor this combination method given underlying normal distributions. We employed kernel density estimation Matlab toolbox [7] for implementation of this method.

For comparison we used generic classifiers provided in PRTools[8] toolbox. SVM is a support vector machine with second order polynomial kernels, Parzen is 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 would simulate sampling score distributions to get training data, training classifiers with this training data and evaluating classifier performance. Since score distributions are available, it is possible to generate arbitrarily large testing set, but instead we simply used formula 3 to numerically get added error. For each setting we average results of 100 simulation runs and take it as average added error. These average added errors are reported in the tables.

In the first experiment (table 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 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 optimal decision boundary will be 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 showed better performance than all other methods, especially for low Bayesian error cases. Only for $STD = .5$ DP method was able to get similar performance.

In the second experiment (table 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 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 gets better faster with increased training data size, but it is not certain. Interestingly, the magnitude of 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 increased number of training samples.

For the third experiment (table 3) we attempted to see how 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. Though additional classifier makes relative combination added error bigger, the

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. Dependence of combination added error on the number of classifiers

dramatic decrease of Bayesian error would be much more important for total error. Also note that result for 3 classifiers and results of first two rows of table 1 have comparable Bayesian errors, with SVM method not performing as well as for two classifiers.

5 Conclusion

In this paper we presented the results of evaluating combination added error. We experimentally showed 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 will easily outperform superior combinator. Thus it is more important what minimum Bayesian error combination has, which is determined by classifiers' performances and their interdependence (assuming that trainable generic classifier is used as combinator and not fixed combination rules, like sum or product rule). The choice of combination algorithm becomes more important when classifiers have small Bayesian errors.

The presented method for combining independent classifiers by means of multiplying one-dimensional densities showed slightly better performance than comparable Parzen classifier. Thus using independence information can be beneficial for density based classifiers. At the same time DP method was still not as good as SVM. It seems that if more training samples were used and more classifiers are combined, DP might be better than SVM.

Though only one type of density functions was used in our experiments, the technique can be easily expanded to other density functions. Clearly, performance of presented methods can be different if other density functions are used. In real-life applications it would make sense to set a hypotheses on available biometric score densities, and perform similar type of experiments in order to find the best combination method.

Still, even if such experiments are performed, and best combination method is found, it is not guaranteed that the combination method will be the best for a particular available training sample. Note that figures in presented tables are averages of added errors over different training sets. In fact there were many simulation cases, where inferior combination algorithm outperformed all other algorithms for a particular training set.

The main motivation of this paper was to find a possibly best combination method for multimodal biometric matchers. Though presented techniques will help to choose a reasonably well performing combination method, 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, these methods might be a bad choice. The ability of density based methods to output posterior class probability can be a decisive factor for their adoption.

References

1. Tumer, K., Ghosh, J.: Linear and order statistics combiners for pattern classification. In Sharkey, A.J., ed.: Combining Artificial Neural Nets: Ensembles and Multi-Net Systems. Springer-Verlag, London (1999) 127–162

2. Kuncheva, L.: A theoretical study on six classifier fusion strategies. Pattern Analysis and Machine Intelligence, IEEE Transactions on **24** (2002) 281–286
3. Fumera, G., Roli, F.: Performance analysis and comparison of linear combiners for classifier fusion. In: SSPR/SPR. (2002) 424–432
4. Fumera, G., Roli, F.: Analysis of error-reject trade-off in linearly combined multiple classifiers. Pattern Recognition **37** (2004) 1245–1265
5. Alexandre, L.A., Campilho, A.C., Kamel, M.: On combining classifiers using sum and product rules. Pattern Recognition Letters **22** (2001) 1283–1289
6. Silverman, B.W.: Density estimation for statistics and data analysis. Chapman and Hall, London (1986)
7. Beardah, C.C., Baxter, M.: The archaeological use of kernel density estimates. Internet Archaeology (1996)
8. Duin, R., Juszczak, P., Paclik, P., Pekalska, E., Ridder, D.d., Tax, D.: Prtools4, a matlab toolbox for pattern recognition (2004)