The Multimodal Technologies usage for Security Systems Design (II): Learning process, data classification and performance evaluation

Sorin Soviany¹, Sorin Pușcoci¹

Abstract: The multimodal technologies include methods and algorithms that are applied to process data from multiple and, typically, independent sources, as support to generate and provide meaningful information (such as scoring) for decisions functions in various applications. The multimodal technologies provide the information support to the decision systems, in this case with focus on data and networks security. The principles and tools for the multimodal security systems design concern relevant issues for the security functions optimization in order to reach the required performances customized for the real applications end-users. The design tools are based on Artificial Intelligence (AI) area specific techniques, particularly on Machine Learning (ML). The performance amounts for the multimodal systems security and effectiveness assessment are defined in the paper, given their practical utility.

Keywords: multimodal technologies, machine learning

1. Introduction

A modern approach for the security systems design and development is based on the multimodal technologies. These technologies include algorithms that are designed to process data from

¹Institutul Național de Studii și Cercetări pentru Comunicații, București, România
multiple and independent sources, outcoming meaningful information (such as scoring) for decision making in various applications including those from data and networks security domain. The design and development of security functions based on multiple factors, with various data types processing, has significant challenges concerning the complexity and performances adjustment even for conventional desktop-based applications [1]. The principles and tools for the multimodal security systems design concern relevant issues for the security functions optimization in order to reach the required performances customized for the real applications end-users. The design tools are based on Artificial Intelligence (AI) area specific techniques, particularly on Machine Learning (ML). The core functionality is represented by the learning process together with some optimizations tools and methods that are used in order to provide high-performance solutions for the real applications.

The paper has the following structure: Section 2, that presents the learning process and modeling principles, also with a bried presentation of the typical data fusion rules; Section 3, that presents the typical performance measures for the multimodal security systems; Section 4, that concludes this paper.

2. The learning process/modeling, optimization and data fusion

The multimodal security systems design requires to perform a learning process based on relevant data in order to develop a high-performance solution that should meet the real security application requirements. Typically a supervised learning process is implemented for biometric applications, but other applications like intrusion detection (IDS, Intrusion Detection Systems) require to develop a combination of supervised and unsupervised models, such as to provide a reliable outliers or anomalies detection before the supervised learning. In the remainder of the paper only the supervised learning (data classification) is considered, together with:

- optimizations methods: starting from the various available models for data classification (supervised learning) that could be applied in order to provide specific security decisions, some particular design options may be considered in order to enhance the overall learning process performance, such as:
  - Moving from multi-class to binary classification;
  - Target-vs.-non-target classification, sometimes within a hierarchical classifier (with multi-stage decisions) for multi-level security systems;
- data fusion rules, at several processing stage. The data fusion is the core functionality for any security system in which several data types should be integrated within the multimodal approach.

2.1 Data classification process

The data classification process in multimodal security systems is performed with 2 complementary approaches [1]:

---

The design and development of security functions based on multiple factors, with various data types processing, has significant challenges concerning the complexity and performances adjustment even for conventional desktop-based applications [1]. The principles and tools for the multimodal security systems design concern relevant issues for the security functions optimization in order to reach the required performances customized for the real applications end-users. The design tools are based on Artificial Intelligence (AI) area specific techniques, particularly on Machine Learning (ML). The core functionality is represented by the learning process together with some optimizations tools and methods that are used in order to provide high-performance solutions for the real applications.

The paper has the following structure: Section 2, that presents the learning process and modeling principles, also with a bried presentation of the typical data fusion rules; Section 3, that presents the typical performance measures for the multimodal security systems; Section 4, that concludes this paper.

2. The learning process/modeling, optimization and data fusion

The multimodal security systems design requires to perform a learning process based on relevant data in order to develop a high-performance solution that should meet the real security application requirements. Typically a supervised learning process is implemented for biometric applications, but other applications like intrusion detection (IDS, Intrusion Detection Systems) require to develop a combination of supervised and unsupervised models, such as to provide a reliable outliers or anomalies detection before the supervised learning. In the remainder of the paper only the supervised learning (data classification) is considered, together with:

- optimizations methods: starting from the various available models for data classification (supervised learning) that could be applied in order to provide specific security decisions, some particular design options may be considered in order to enhance the overall learning process performance, such as:
  - Moving from multi-class to binary classification;
  - Target-vs.-non-target classification, sometimes within a hierarchical classifier (with multi-stage decisions) for multi-level security systems;
- data fusion rules, at several processing stage. The data fusion is the core functionality for any security system in which several data types should be integrated within the multimodal approach.

2.1 Data classification process

The data classification process in multimodal security systems is performed with 2 complementary approaches [1]:
• the statistical approach with supervised learning, in which the processing output is an estimator of the posterior probability for the class membership or, depending on the applied model structure, the distance to a decision boundary. This value is computed or estimated using a classifier’s discriminant function;

• the deterministic approach with distances computation and matching score, in which the similarity degree is evaluated between a current pattern and a reference one (both of them being represented as feature vectors), using a point-to-point comparison (or, if case, with a distance estimation).

2.1.1 The supervised learning for data classification [1]

The supervised learning uses classification models that are varying depending on the following criteria [1],[2],[3],[4],[5],[6]:

• the features representation or combination modality, criterion according to which the available classifiers are grouped in 2 categories [2]:
  ➢ linear classifiers, in which the decision is based on a classifier output that is computed using a linear combination of the input features;
  ➢ non-linear classifiers, that generate class membership decisions using non-linear combinations of the input features;

• the classifiers parameters and hyper-parameters evaluation modality, that differentiates the classifiers into the following categories [2],[4],[5]:
  ➢ generative classification models, that directly uses class-conditioned probability density functions \( p(x|\text{Class}) \) where \( x \) is the input feature vector and \( \text{Class} \) is the class label of the current sample). The decision rule derives from Bayes theorem application, leading to models that are different depending on the relationships between the class covariance matrices and some specific parameterizations. Actually the generative models differ according to the parameterization degree [4]:
    o generative classifiers with parametric models, in which the underlying models (typically Gaussians) design uses statistical parameters including class means and variances (covariances), for Gaussian class distributions. The Gaussian parametric models-based classifiers are NMC (the Nearest Mean Classifier), LDC (the Linear Discriminant Classifier), QDC (the Quadratic Discriminant Classifier with normal densities), GMM (Gaussian Mixture Models);
    o non-parametric generative classifiers, that do not use explicitly parameterized models with statistical estimates of the input data. The non-parametric classifiers are the Parzen classifier and the Naive-Bayes classifier. Actually a certain parameterization still remains (related to the underlying model mathematical structure), but these parameters do not represent statistical properties of the input samples (statistics from the class distributions);
Discriminative classification models, that uses direct discrimination rules in order to maximize the output quality on a training dataset. The discriminative classifiers directly model the class posterior probabilities or the decision function, without the explicit usage of per-class probability density functions estimates. The discrimination criterion allows to distinguish between 2 types of classification models:

- **Distance-based discriminative classifiers**, in which the discriminant estimation is done using a distance measure. The most common distance-based classifiers are the Fisher classifier, the logistic classifier, the KNN classifier (K-Nearest Neighbors decision rule), the SVM (Support Vector Machines) classifier;
- **Discriminative classifiers for the error minimization** performs the discrimination function using an optimal criterion for the error rate minimization. The most common classifiers based on discrimination for error minimization are: the perceptron, the multi-layer neural networks and the decision tree;

- **The classes number** separates 2 types of classifiers:
  - **Binary classifiers**, that only discriminates between 2 classes;
  - **Multi-class classifiers**, in which the separation is among more than 2 classes.

### 2.1.2 The matching (similarity) score computing for data classification [1]

The deterministic approach using matching or similarity scores computes a distance measure within the feature vector space; the distance is evaluated between the current pattern and a reference sample that was previously enrolled. This approach is typically applied for many biometric systems in which the design is based on the comparison between the current biometric sample applied to the system input and one or several biometric templates retrieved from the system database.

This approach proceeds to compute the similarity score $S_0$ between the current feature vector $x$ and the reference $y$ using a certain distance. The procedure requires the following steps:

- **The distance computation** $d(x, y)$. The typical distances for this computational process are Euclidian distance and Mahalanobis distance [5],[7]:
  - The Euclidian distance is commonly used to compute a distance-based score within the input feature space. For 2 vectors $x$ and $y$ with the same size $D$, the Euclidian distance is computed as follows:
  $$d_E(x, y) = \sqrt{\sum_{i=1}^{D} (x[i] - y[i])^2} \tag{1}$$
  - The Mahalanobis distance is a distance based on the correlation among variables and can be used to evaluate the similarity degree among 2 variables sets. Its definition for 2 vectors $x$ and $y$ with the same size $D$ is the following:
  $$d_M(x, y) = \sqrt{(x - y)^T \cdot \Sigma^{-1} \cdot (x - y)} \tag{2}$$

Where $\Sigma$ is the covariance matrix of the 2 feature vectors, computed as
\[
\Sigma = \text{Cov}(x, y) = E \left[ (x - E[x]) \cdot (y - E[y])^T \right] \quad (3)
\]

[] is the operator that provides the mean of a random variable. For a binary classification problem (such as the biometric verification with authentic vs. impostor discrimination or the fraudulent actions detection with genuine vs. attack discrimination), the covariance matrix is typically computed with the following equation [5]:

\[
\Sigma = P_1 \cdot \Sigma_1 + (1 - P_1) \cdot \Sigma_2 \quad (4)
\]

in which \( \Sigma_1 \) and \( \Sigma_2 \) are the covariance matrices for the 2 classes, and the weights are their priors. The 1st class is usually the target class for the application. The Mahalanobis distance has 2 properties that make it a very useful similarity criterion [8]:

- **exploiting the existing correlation among the features.** This property support some additional decisions optimizations because it allows to use only the most informative features, during the learning process (supervised classification), usually by discarding the most correlated features;
- **scaling-invariance property.** that allows to apply further scaling transforms on the input features, such as to prevent chancing the order relationships among the computed features, distances or similarity scores;

**the distance score normalization** is a process that performs the individual scores conversion to a common numerical values range, usually just before their combination with a matching score-level fusion rule. This computational task addresses the following problems [7]:

- the scoring heterogeneity (distance or similarity scores, various functions used to compute the required scores);
- the variety of the numerical ranges in the scoring computation;
- the different statistical distributions of the computed scores.

The scoring normalization is usually applied with one of the following 2 approaches [7]:

- the **still score normalization**, in which the parameters used for normalization procedure are fixed using a given reference dataset;
- the **adaptive score normalization**, in which the normalization parameters are estimated using the current feature vector, allowing to adapt the result to some variations of the input data[3],[7],[9].

The common normalization techniques for the similarity scores include [3],[7]:

- **Min-Max normalization**, with the model given by

\[
S_{i,\text{norm}} = \frac{S_i - S_{\text{min}}}{S_{\text{max}} - S_{\text{min}}}, i = 1, n \quad (5)
\]

in which the minimum and maximum bounds of the individual score are given: \( S_{\text{min}} \) and \( S_{\text{max}} \), respectively. \( n \) is the overall individual scores number that are computed; the scores set is \( S = \{ S_i = d_i(x_i, y_i) | i = 1, n \} \), where the scoring computation \( S_i \) is based on
a certain distance measure within the feature vector space, for example Euclidean or Mahalanobis distance. A suitable scaling transform could be found in order to properly adjust the scores bounds, for example to provide settings as: \( S_{\min} = 0 \) and \( S_{\max} = 1 \).

This normalization technique is sensitive to the outliers within the input data;

- **Decimal scaling normalization**, computed with the model given by
  \[
  S_{i,\text{norm}} = \frac{S_i}{10^m}, \quad i = 1, n
  \]
  in which \( n \) is the number of computed scores, and
  \[
  m = \log_{10} \max_{i=1,n} S_i
  \]
  This technique is suitable when the matching provides logarithmic-scale scores;

- **z-score normalization**: is the most common normalization technique for matching scores. It uses some statistics drawn from the computed scores, typically means and standard deviations. This is suitable especially when prior information are available as concerning the scores statistic distribution. The normalized score is computed using the following equation
  \[
  S_{i,\text{norm}} = \frac{S_i - \mu_S}{\sigma_S}
  \]
  where \( \mu_S \) and \( \sigma_S \) are the mean and standard deviation, respectively, for the computed matching scores. The main challenge is that the scores mean and standard deviation are sensitive to the input data outliers. Also this technique does not guarantee a common numerical range for the transformed scores. If the scores distribution does not follow the Gaussian law, then the z-score normalization does not preserve the initial scores distribution. Actually the mean and standard deviation are optimal location and scale parameters only for Gaussian distributions;

- **MAD (Median Absolute Deviation) normalization**: a normalization scheme that is based on parameters that are not sensitive to the input data outliers nor to the extreme values from the computed scores distributions. This normalization model is given by
  \[
  S_{i,\text{norm}} = \frac{S_i - \text{median}}{\text{MAD}} = \frac{S_i - \text{median}}{\text{median}(|S_i - \text{median}|)}
  \]
  This normalization scheme does not preserve the original scores distribution type;

- **Normalization with sigmoid functions**: it is a normalization scheme that transforms the matching scores such as the output values belong to the range \([0,1]\). The sigmoid function, given by
  \[
  f(x) = \frac{1}{1 + \exp(-x)}, \quad f: \mathbb{R} \rightarrow [0,1]
  \]
  provides an output value between 0 and 1 for any real input value. The more general model of the sigmoid-based normalization is computed with the parametric equation
\[ S_{i,\text{norm}} = \frac{1}{1 + A_i \cdot \exp(-\alpha_i \cdot S_i - \beta_i)}, i = 1, n \] (11)

in which the values of parameters \( A_i, \alpha_i \) and \( \beta_i \) are provided based on the experimental or training datasets that are used for the system design, and \( S_i = d_i(x_i, y_i) \). The normalization with sigmoid functions allows [7]:

- to compare the distance-based matching scores;
- to transform the distance-based scores into similarity scores;
- to match the similarity scores against the classification outputs (issued as posterior probabilities).

The normalization of the distance-based scores could be done using the simple sigmoid function or the double sigmoid function [3],[7],[10]:

- the distances normalization with the simple sigmoid function is given by

\[ S_{i,\text{norm}}(x_i, y_i) = \frac{1}{1 + A_i \cdot \exp(-\alpha_i \cdot d_i(x_i, y_i) - \beta_i)}, i = 1, n \] (12)

where the coefficients \( A_i, \alpha_i \) (shape parameter) and \( \beta_i \) (offset) result from the reference or experimental available data for each of the feature vectors;

- the distances normalization with the double sigmoid function is given by [7],[9]

\[
S_{i,\text{norm}}(x_i, y_i) = \begin{cases} 
1, & \text{if } d_i(x_i, y_i) < \theta \\
1 + A_i \cdot \exp\left(-B_i \cdot \left(\frac{d_i(x_i, y_i) - \theta}{C_{1,i}}\right)\right), & \text{if } d_i(x_i, y_i) \geq \theta 
\end{cases} \] (13)

in which:
- the coefficients \( A_i \) and \( B_i \) are fixed based on the experimental data. These amounts represent the shape parameters of the sigmoid function;
- the coefficients \( C_{1,i} \) and \( C_{2,i} \) (scaling factors) define the boundaries of the regions in which the sigmoid function exhibits an almost linear behavior [2];
- \( \theta \) is a threshold that relates to the security degree that should be provided through the design.

Looking to the equations (12) and (13), one can see the following relationships among the shape (\( A_i, \alpha_i \) and \( B_i \)), scaling (\( C_{1,i}, C_{2,i} \)) and offset (\( \theta \)) coefficients:

\[
\alpha_i = \frac{B_i}{C_{y,i}}, y \in \{1, 2\}, i = 1, n \] (14)

\[
\beta_i = \frac{B_i}{C_{y,i}} \cdot \theta, y \in \{1, 2\}, i = 1, n \] (15)

The normalized distance score \( S_{i,\text{norm}}(x_i, y_i) \) evaluates the difference between the currently testing feature vector \( x_i \) and the reference \( y_i \).
• **the conversion from distance score to similarity score.** The similarity score evaluates the closeness between the matched feature vectors. A higher similarity score value shows an increased number of similar feature points. The similarity evaluation between the current feature vector and the reference requires the conversion from distance to similarity score. For distance scores with sigmoid-based normalization, with values within the range [0,1], the conversion from distance to similarity could be done with the following equation:

\[ S_{i,0} = s(x_i, y_i) = 1 - S_{i,norm}(x_i, y_i), i = 1, n \]  

(16)

If the distances are normalized with other techniques, providing different output ranges \([S_{min}, S_{max}]\), in which usually \(S_{max} > 1\) (with a value that is fixed according to the scoring application objectives), but typically the lowest bound remaining 0, the conversion could be done with the model

\[ S_{i,0} = s(x_i, y_i) = S_{max} - S_{i,norm}(x_i, y_i), i = 1, n \]  

(17)

• **finding a relationship between the similarity score and the posterior probability.** The Verlinde model defines a relationship between the classifiers outputs (given as posterior probabilities) and the matching scores (that evaluates the similarity degree). This model provides an error function for the class membership decision, given the current vector \(x\), as a relationship between the similarity score \(S_{i,0}\) and the class posterior probability \(P(\text{Class} \mid x)\), where \(\text{Class}\) is the class label for the target class, depending on the application [11]:

\[ \varepsilon_{0,j}(x_i) = \left| P(\text{Class}_j \mid x) - S_{i,0} \right|, i = 1, n, j = 1, C \]  

(18)

where \(C\) is the classes number, \(n\) is the count of the integrated data types, \(S_{i,0}\) is the similarity score computed using equations (16) or (17). Actually the last equation (24) combines the 2 types of outputs: the deterministic output based on similarity score resulted from a matching process and the probabilistic score computed by a discriminant function \(g\) representing the underlying model of a classifier (the numerical output of a predictive model, as in typical machine learning supervised). This relationship modeling, given by eq. (18), is only reliable if the classifier output, a predictor of a class membership for the testing feature vector \(x\), given as \(\text{Class} = g(x)\), meets one of the following conditions:

- the classifier output is a class posterior probability, estimated for a given input feature vector, using Bayesian decision rules, with specific parameterizations and particular conditions for the covariance matrices

\[ g(x) = P(\text{Class}_j \mid x), j = 1, C \]  

(19)

- the classifier output is the value of a discriminant function, but typically normalized such as to convert the discriminant value to a value within the range \([0,1]\), allowing the provided output to evaluate the class membership degree:

\[ g^*(x) = \text{sigm}(g(x)) = \frac{1}{1 + A \cdot \exp(-B \cdot g(x))} \]  

(20)
where the coefficients $A$ and $B$ are, as for the equations (11)-(13), the shape parameters of the sigmoid function $g^*$ that should be applied on the classifier discriminant function output, $g(x)$.

There are some typical tools and methodological approaches that are currently applied to design high-performance security systems based on various machine learning modeling techniques. The following techniques, methods and tools are typically used to design and evaluate multimodal security systems based on machine learning approaches [1]:

- **the learning curves representation** to find out the optimal learning models with the properly fixed training samples per class. The learning curve for a classifier is a **graphic representation of some performance measures** (typically the average classification error rate for all the considered classes) **in respect to the available training set size** (the number of training samples per class). The learning curve allows to properly fix the training dataset size and also to compare several classification models for various sizes of the available training dataset. An example of learning curve representation for several models that run on a difficult dataset for a fingerprint recognition system is given in figure 1. A difficult dataset

![Learning curves for several classifiers on difficult dataset](image1.png)

**Figure 1:** Learning curves for several classifiers on difficult dataset

is one in which the separation within the classes if difficult, sometimes because of an intra-class variance that significantly exceed the overall inter-class variance. The learning curve for a classifier shows that an **ideal model** has a performance that asymptotically is going to the **minimum classification error rate** (**Bayes error** [2],[5],[12]), as depicted in figure 2.

![Learning curves for the optimal (ideal) classifier: the asymptotic generalization and apparent error rates](image2.png)

**Figure 2:** Learning curves for the optimal (ideal) classifier: the asymptotic generalization and apparent error rates [1],[12]
This theoretical trend is only true for training datasets with many examples, or if the original data distribution is full known. The **apparent errors** are achieved on the **training subset**, while the **generalization errors** are achieved on the **independent test (validation) subset**. The Bayes error for a classifier depends on the input data representation (feature space). A **non-ideal model** provides a sub-optimal performance level. The learning curves for **ideal** and **non-ideal** models are represented in figure 3 (actually a generic representation).

---

**Figure 3:** Learning curves for the **ideal** and **non-ideal classifier** [1],[12]

- **the classifier bias**, for a given training set size, is defined as the **difference between the generalization and the apparent error** rates [2],[12]:
  \[ \Delta \varepsilon = \varepsilon - \varepsilon_A \]  
  \[ (21) \]
  where \( \varepsilon \) is the **generalization error** rate and \( \varepsilon_A \) is the **apparent error** rate;

- **the classifier complexity addressing**, while considering the training sets and feature spaces dimensionalities, in order to avoid or minimize the **peaking, the classifier performance bounding when reaching certain thresholds for those amounts**. The complexity of a classifier is given by [2],[12]:
  - **the number of the independent parameters for the underlying discriminant model**, with impact on the model **generalization** power;
  - **the feature space dimensionality**, with impact on the class covariances computation complexity;
  - **the number of operations required to compute the class covariances**;
  - **the local properties of the input data**, with impact on the decision boundary complexity;

- **the confusion matrix**, a 2-dimensional array representation of the classification decisions, that are matched against the true class membership information belonging to the input data [12]:
  \[ CM \left( \text{Class}_i, \text{Class}_j \right) = N_{i,j}, i,j = 1,C \]  
  \[ (22) \]
where $N_{i,j}$ is the number of the examples belonging to the true class $C_i$ while the classifier decision is for the class $C_j$. The confusion matrix rows contain the true class labels for the input samples (as row indices); the columns index the classifier output decisions on the input data. The confusion matrix entries count the matching between the true class labels and the classifiers decisions, for each of the specified classes. The diagonal entries store the right classifiers decisions while the non-diagonal entries store the wrong decisions about class membership;

- **the usage of cross-validation techniques** in order to enhance the generalization power of the designed model, such as to prevent or reduce the over-fitting of training data [4]. The cross-validation is *a classifier design and evaluation technique that looks to improve its generalization performance by running several rounds of training and validation, with varying the used datasets*. Within each round, the design dataset is partitioned into complementary subsets; the classifier training is done on one subset, and the validation is performed on the remaining subset or examples that were not used for training. Several rounds are run with different partitioning of the design dataset [12];

- **the classes unbalancing approaching** as concerning *the number of training samplers per class and the classes representation within the training dataset*;

- **the classifier selection methodology**, with the following steps [4],[12]:
  1. providing the relevant datasets for the security system design and testing;
  2. starting the design (training/learning) process with the lowest complexity models (e.g. NMC, Fisher and LDC). These models provides good performances when the training dataset contains a small number of samples per class;
  3. testing the models in their increasing complexity order;
  4. comparing the performances on training and testing (validation) datasets, using the learning curves.

The selection and evaluation of an optimal model for the application goals (the desired security performance) require to take in account the peaking. This means that the generalization power of the model is constrained by the ratio between the Training set Size ($TS$)-the number of training samples per class- and the problem complexity given by the Feature set Size ($FS$)-the number of used features. A required condition for a good generalization performance is the following [2],[6]: maximization of the ratio $\frac{TS}{FS}$. The peaking is depicted in figure 4 [1],[2]. The peaking means that for a given training set size ($TS$) and with a significant increasing of the feature space size ($FS$), *the classification errors rate reduction is only achieved till reaching a certain boundary that constrains the overall performance of the designed system*. Therefore, if the provided features number $FS$ exceeds a certain threshold $FS_0$, the performance trend reverses and it does not more happen what it was normally expected for an extended feature set; therefore in this case the classification errors rate actually starts to increase. From figure 4 one can see that the critical threshold of the features number (parameter $FS$) moves to the right of the
horizontal axis, therefore towards the higher values with the increasing of the training samples number. A minimum value for this threshold is reach according to \[ 2 \]

$$ F S^* = \frac{TS}{\alpha} $$

(23)

where the coefficient \( \alpha \) must meet the bounding condition \( \alpha \in [2,10] \), that is typically for the best performances.

### 2.2 The learning (modeling) process optimization

The learning or modeling process could be significantly improved to provide high performance according to the real security application if some design options are applied. These functional options include the following technical approaches:

- Moving from multi-class to binary classification;
- Applying a target-vs.-non-target classification, typically within a hierarchical classifier in order to design multi-level security systems.

a) Transition from multi-class to binary classification

The typical methods that could be applied in order to build multi-class models using several binary classification models are the following [4],[13],[14]:

- One-vs.-all others method;
- One-vs.-one method;
- DDAG (Decision Directed Acyclic Graph)-based method.

b) Target vs. non-target classification and hierarchical classification

It is a design approach that is suitable for multi-level security applications in which the end-users have various roles and authorization degrees. The design is based on 2 classification techniques:

- Detector classifiers, that are trained to separate between a target and a non-target class;
- Discriminant classifiers, in which the training process is done for all classes.
These 2 models could be integrated in hierarchical classifications systems with several decision stages. An example of the hierarchical classifier for iris recognition is shown in figure 5 [15].

![Hierarchical Classifier for Iris Recognition](image)

Figure 5: A hierarchical classifier for iris recognition [23]

### 2.3 Data fusion

The **data fusion** is the functional core of any multimodal security system. The data (multimodal) fusion is the combination process that applies on data that are achieved from several sources but concerning the same object or process. Depending on the fusion level, the results could be [1]:

- a combined set of features, for the case of low-level fusion (particularly feature-level fusion);
- a global matching (similarity) score, that is computed based on several individual scores, for the case of high-level fusion (score-level fusion);
- a final decision that is dependend on the individual decisions issued by each classifier, for the case of high-level fusion (decision-level fusion).

Depending on the raw data processing stage in which the combination has to be applied, there are 2 main categories of multimodal fusion [7]:

- **low-level fusion**, that includes pre-classification fusion schemes such as sensor-level fusion and feature-level fusion. The most common one is the feature-level fusion, that can be implemented using the following 2 variants [7],[10]:
- **concatenation-based fusion**, typical for heterogeneous feature vectors (with different sizes);
- **functional combination fusion**, that could be applied for feature vectors with the same sizes.

- **high-level fusion**, that includes **post-classification fusion** schemes such as **score-level fusion**, **rank-level fusion** and **decision-level fusion**. The score-level fusion is the most implemented one, typically based on several mathematical rules for scores combination (for example sum rule, average rule, min-rule). Some of these fusion rules could be provided with parameters such as various weigths.

### 3. Performance measures for the security systems based on multimodal technologies

The overall design and development process of security systems using multimodal technologies requires a careful consideration of the performance issues, in order to meet the real application constraints.

Under this methodological framework, and taking into account the typical functional components of any multimodal security system, a classifier performance is evaluated using the following approaches [16]:

- the **training performance**, that evaluates the classifier capability to make right decisions on the class membership for the training samples (therefore the **training optimization**);
- the **generalization performance**, that evaluates the classifier capability to identify the class membership for the new (unseen) examples.

The classification errors are usually caused by the **classes overlappings within the feature space**, as depicted in figure 6 [16].

![The classes overlapping within the feature space (2D example)](image)

Figure 6: The classes overlapping within the feature space (2D example) [1],[16]
The typical reasons of the classes overlapping within the feature space are the following [1],[2],[5],[16]:

- a poor feature space dimensionality, that is not enough to cover the overall set of attributes;
- the low relevance of the extracted features, that do not describe the most useful discriminant properties;
- the noise and outliers within the input datasets.

The security performances of the multimodal systems are evaluated using amounts that relate to the positive (P)/negative (N) outputs of the predictive model. The following amounts are typically used to define the security performances [1],[12]:

- **TP (True Positive):** the count of right decisions for a target class (the positive class for the application). A true positive decision is generated when the predicted value $P$ is the same as the real (true) value;
- **TN (True Negative):** the count of right decisions for a non-target class (the negative class). A true negative decision is generated when both values (predicted and real) are $N$;
- **FP (False Positive, Type I Error):** the count of wrong decisions for the target (positive) class. A false positive decision (false alarm/alert) is generated when the predicted value is $P$ but the real value is $N$;
- **FN (False Negative, Type II Error):** the count of wrong decisions for the non-target (negative) class. A false negative decision is generated when the predictor output is $N$ but the real value is $P$.

The following security performances are computed based on the above amounts [1],[12]:

- **TPR (True Positive Rate, sensitivity, detection rate/probability),** given by
  \[
  TPR = \frac{TP}{TP + FN}
  \]  
  \[
  (24)
  \]
- **TNR (True Negative Rate, specificity),** given by
  \[
  TNR = \frac{TN}{FP + TN}
  \]  
  \[
  (25)
  \]
- **FPR (False Positive Rate, false alarms/alerts rate/probability),** given by
  \[
  FPR = \frac{FP}{#N} = \frac{FP}{FP + TN} = 1 - TNR
  \]  
  \[
  (26)
  \]
  where $#N$ is the total amount of decisions for the non-target class;
- **FNR (False Negative Rate),** given by
  \[
  FNR = \frac{FN}{#P} = \frac{FN}{FN + TP} = 1 - TPR
  \]  
  \[
  (27)
  \]
  where $#P$ is the total amount of decisions for the target class
The ROC (Receiver Operating Characteristic) is a graphical representation of the security performances (typically TPR vs. FPR or TNR vs. FNR) for some discrimination threshold variation. A ROC curve contains a set of operating points that are achieved for some security thresholds [1],[2],[12]. An example of ROC representation is shown in figure 7 [17].

![Example of ROC curves for a multimodal system with hierarchical classifier](image)

Figure 7: The ROC curves for a multimodal system with hierarchical classifier [17]

4. Conclusions

In this paper the methodological framework for the multimodal security systems design and performance evaluation is presented. Here the main focus is on the modeling process with supervised learning and the performance evaluation.

The presented framework includes some relevant design tools that are commonly used to develop high-performance systems that must be optimized to meet the real applications constraints, taking in account the performance issues but also the cost and complexity matters. The learning methodology could be applied for many real security applications design and development, such as biometric authentication systems or intrusion detection. However, the potential applications domain is significantly enlarging.

Some design optimization options are also presented, typically for multi-level security applications in which the overall modeling process focuses to provide decisions on the most important class. It is the case of biometric authentication systems in which the recognition should be more accurate for certain individuals with the highest authorization degrees. Finally, the common performance measures are defined and explained; these amounts (typically TPR and
FPR) are used to optimize the designed system accuracy according to the customer application requirements. The application key target indicators (detection rate vs. alert rate) are based just on these amounts (TPR and FPR, respectively). The real application goal is to maximize the detection rate (TPR) for a certain alert rate (FPR) that should be as low as possible.

References

Empirical Study, Springer-Verlag Berlin Heidelberg 2005
http://www.keerthis.com/multiclass_mcs_kaibo_05.pdf


