

# A New Classifier Simulator for Evaluating Parallel Combination Methods

H. Zouari<sup>1,2</sup>, L. Heutte<sup>1</sup>, Y. Lecourtier<sup>1</sup>, A. Alimi<sup>2</sup>

<sup>1</sup> *Laboratoire Perception, Systèmes, Information (PSI), Université de Rouen, France*

<sup>2</sup> *Groupe de Recherche sur les Machines Intelligentes (REGIM), Université de Sfax, Tunisie  
{Hela.Khoufi, Laurent.Heutte}@univ-rouen.fr*

## Abstract

*The use of artificial outputs generated by a classifier simulator has recently emerged as a new trend to provide an underlying evaluation of classifier combination methods. In this paper, we propose a new method for the artificial generation of classifier outputs based on additional parameters which provide sufficient diversity to simulate, for a problem of any number of classes and any type of output, any classifier performance. This is achieved through a two-step algorithm which first builds a confusion matrix according to desired behaviour and secondly generates, from this confusion matrix, outputs of any specified type. We provide the detailed algorithms and constraints to respect for the construction of the matrix and the generation of outputs. We illustrate on a small example the usefulness of the classifier simulator.*

## 1. Introduction

One of the problems that are to be faced in designing a multiple classifier system is the choice of a suitable combination method among the set of existing ones for a given problem of classification [1, 2, 4, 8, 10, 12]. Usual experiments consist in testing exhaustively the possible combination methods that can be used and retain the best one according to some criteria. However the obtained results depend closely on the specific application which has served the experimental proof. On the other hand, there do exist some interesting works in which the combination methods are evaluated on real data sets [3, 4, 7, 9, 10]. But, it is a fact that the use of real data can not yield sufficient variability in classifier performance to evaluate in a deeper manner the combination methods.

Very recently, the use of artificial data (outputs of classifier) generated by a classifier simulator has emerged to address the problem of the evaluation of classifier combination methods [5, 6, 7]. In [5] for example, the author proposes a comprehensive study of the random generation of dependent classifiers to evaluate class type combination methods especially the majority voting. She derives formulas according to how two classifiers can be generated with specified

accuracies and dependencies between them. Based on these formulas, the author proposes an algorithm for generating multiple dependent classifiers. Lecce [6] has also studied the influence of correlation among classifiers to evaluate combination methods of class type. The performance of a combination method is measured on several sets of classifiers, each set differing from the others in terms of recognition rate and level of correlation. Each classifier is simulated at output level and a similarity index is used to estimate the stochastic correlation among the classifiers of each set. Parker [7] proposed a system based on two steps of simulation for evaluating ranking combination methods. The first step of simulation consists of a process for generating confusion matrices from fixed global recognition rates. The second simulation step consists of a function that uses the probabilities derived in this confusion matrix to generate outputs which can be correct, incorrect or rejected. The proposed method has also been used to evaluate combination methods of class type. Although interesting, two drawbacks can be reproached to the previous works: the first one is that the studies are rather limited since the classifier simulators used aimed at simply generating abstract-level type outputs and are thus not applicable to measurement type combination methods; the second one is that the behaviour of the developed classifier simulators is only controlled by one global parameter (recognition rate).

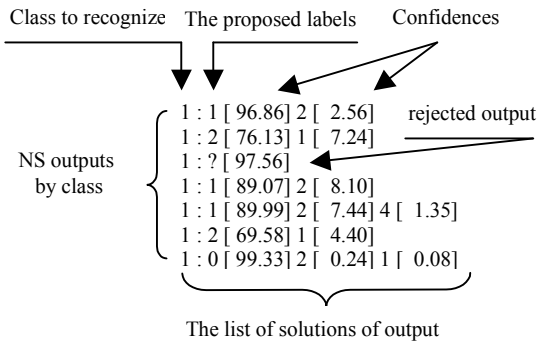
Building a classifier simulator that exhibits sufficient variability to move nearer a real behaviour needs to be able to control other parameters. To this end, we propose a classifier simulator which is able to generate automatically as many lists of outputs as desired according to a user-specified behaviour. Now, the classical way to evaluate the performance of a classifier is to measure on a test set of outputs provided by the classifier some global performance such as the overall recognition rate and/or rejection rate. More precise analysis of the classifier performance can be measured through a confusion matrix which highlights the behaviour of the classifier on each class. Simulating a classifier is the inverse operation: starting from input parameters such as overall recognition or rejection rates which represent the desired behaviour we want to simulate, we build a confusion matrix to fix the behaviour of the classifier for each class then we

generate, according to this confusion matrix, as many lists of outputs as desired. To fit the reality (i.e. simulate any classifier behaviour), we must thus manage as input parameters the number of classes, the recognition and rejection rates and their standard deviations (to fix each class behaviour) but also and at the same time the maximum number of solutions in output lists and the corresponding recognition rates within the lists of solutions.

The paper is organized as follows. Section 2 describes the methodologies in developing the classifier simulator and the different parameters (desired behaviour) used for this issue. Section 3 presents the construction of the confusion matrix according to these parameters and the main constraints to respect for this construction; Section 4 presents the algorithm for the generation of outputs. Simple experimental results are reported in section 5 to prove the efficiency of the simulator. Conclusion is drawn in section 6.

## 2. Proposed method

The aim of the proposed classifier simulator is to generate randomly measurement type outputs i.e. a confidence value is associated to each solution proposed by the classifier (an example is shown in figure 1). Note that we have chosen to generate this type of outputs because it can also be easily transformed into ranked outputs (by “forgetting” confidence values) or into class-label outputs (by retaining only the first solution in the list).



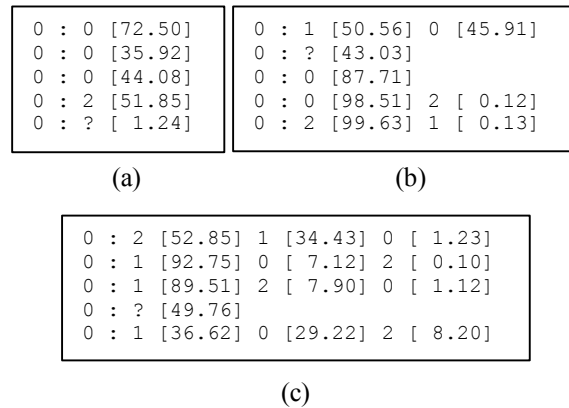
**Figure 1: Description of the simulator outputs**

The inputs of the classifier simulator are parameters which allow to define the classification problem and the global performance of the classifier. These parameters are:

**The number N of classes:** This parameter enables to simulate any pattern recognition problem. We could for example simulate a 2-class problem or a handwritten digit recognition problem (N=10) as well as a handwritten Korean character recognition problem (N=3755) [11].

**The number NS of outputs:** This parameter will be used to fix the number of outputs to be generated for each class by the simulator (same number for all the classes).

**The number K of maximal labels:** It corresponds to the length of the solution list that will be generated for each output. K may vary from 1 to N : for K=1, the simulator generates only one solution for each output (fig. 2a); for K between 2 and N-1, at most K solutions are generated except in the case of a rejected output for which there is only one label “?” (fig. 2b); for K=N, each output contains N labels excepted for the rejected outputs (fig. 2c).



**Figure 2: Different types of outputs generated by the classifier simulator for a 3 class problem (a) with K=1; (b) with K=2; (c) with K=3.**

**Mean recognition rate  $TL^K$ :** It stands for the overall recognition rate in the K first solutions that is the ratio of the number of outputs for which the true class appears among the K first solutions on the total number of outputs.

**Mean rejection rate  $TR^K$ :** It stands for the overall rejection rate that is the ratio of the number of rejected outputs on the total number of outputs. Of course,  $TR^1 = \dots = TR^K = \dots = TR^N$  since a rejected output contains only one label.

Note that we do not include as input parameter the confusion rate  $TC^K$  that would represent the ratio of the number of outputs for which the true class does not appear at all in the K first solutions on the total number of outputs. Indeed,  $TL^K$ ,  $TR^K$  and  $TC^K$  are linked with the relation :  $TL^K + TC^K + TR^K = 100\%$ .

$TL^K$  and  $TR^K$  enable to control the global performance of the simulated classifier. The classifier behaviour will be completely determined through the confusion matrix  $CM^K$ . Note that formally speaking,  $CM^K$  can be called “confusion matrix” only for K=1. For K varying from 2 to N, we would rather have to call it “matrix of presence”. The matrix  $CM^K$  consists of N rows and N+1 columns, where N denotes the number of classes and the column (N+1) the rejection column. The diagonal

elements  $TL_i^k$  are the top K recognition rate per class (i.e. the ratio of the number of outputs labelled i for which the label i appears in the K first solutions on the number of outputs labelled i). Their mean gives the mean recognition rate  $TL^K$ . The off-diagonal elements  $TC_{i,j}^k$  (confusion rates) represent the ratio of the number of outputs labelled i for which a solution labelled j ( $j \neq i$ ) appears in the K first solutions. Note that for  $K=1$ ,  $TC_{i,j}^k$  is a true confusion rate between class i and class j. Otherwise, it is rather a co-occurrence rate. The other elements  $TR_i^k$  (in the column N+1) represent the ratio of rejected outputs for each class i. The mean of these rates is  $TR^K$ .

**Standard deviations  $\alpha^K$  and  $\beta^K$ :** These two parameters will just allow to fill the matrix  $CM^K$ . They enable to introduce sufficient variability in the classifier behaviour while controlling this variability.  $\alpha^K$  (respectively  $\beta^K$ ) will be used to fix the range of variation of  $TL_i^k$  (resp.  $TR_i^k$ ), i.e. each  $TL_i^k$  (resp.  $TR_i^k$ ) will be randomly drawn in the range  $[TL_i^k - \alpha^K; TL_i^k + \alpha^K]$  (resp.  $[TR_i^k - \beta^K; TR_i^k + \beta^K]$ ).

Now, the role of the simulator is to generate, for each class, NS outputs with a list of K solutions. The generation of outputs respects a specified performance in the K first solutions and is realized in two steps:

- Building of the matrix  $CM^K$  according to  $TL^K$ ,  $\alpha^K$ ,  $TR^K$ ,  $\beta^K$ .
- Generation of the list of K solutions at most for each output using the matrix  $CM^K$  and association of a confidence value to each solution.

Now that the involved parameters are defined, we describe in the following section the two-step algorithm which first builds the matrix  $CM^K$  according to  $TL^K$ ,  $\alpha^K$ ,  $TR^K$ ,  $\beta^K$  and secondly generates, for each class, NS outputs with a list of K solutions using the matrix  $CM^K$  and association of a confidence value to each solution. The generation of outputs respects a top K behaviour (i.e. specified performance in the K first solutions).

### 3. Building of the matrix $CM^K$

In order to illustrate the different constraints that must be respected, consider first the outputs of class 0 generated previously in figure 2. The list of outputs generated in figure 2a contains three outputs recognized as 0, one confused as 2 and one rejected. This corresponds to a recognition rate of 60%, a confusion rate of 20% and a rejection rate of 20%. Consequently, to generate type I outputs ( $K=1$ ), we may use the confusion matrix  $CM^1$  in which the sum of rates for each class i (row) must be equal to 100%. Then, the

construction of  $CM^1$  consists in respecting the following constraint :

$$TL_i^1 + \sum_{j=1, i \neq j}^N TC_{i,j}^1 + TR_i^1 = 100\% \quad (1)$$

In figure 3c where  $K=N$ , the total number of solutions generated for each class is equal to  $N*NS$ . This corresponds in the matrix  $CM^N$  to the sum of rates that is equal to  $N*100\%$ . As the rejected outputs are composed of only one solution, we must multiply the rejection rate by N. Then, to generate this type of outputs, we must construct the matrix  $CM^N$  respecting the following constraint for each class i :

$$TL_i^N + \sum_{j=1, i \neq j}^N TC_{i,j}^N + N*TR_i^N = N*100\% \quad (2)$$

In figure 3b, the list of solutions can vary from 1 to K for each output. From the example, we can establish that 60% of outputs include the true class "0" in the top two solutions. Likewise 40% of outputs include the label "1", 40% of outputs include the class label "2" in the top two solutions and 20% of outputs are rejected. In this case, the sum of rates for each class can be lower or at most equal to  $K*100\%$ . For this reason, we must respect the constraint (3) to construct the matrix  $CM^K$ :

$$TL_i^K + \sum_{j=1, i \neq j}^N TC_{i,j}^K + K*TR_i^K \leq K*100\% \quad (3)$$

The filling of the matrix  $CM^K$  begins with the generation of the rejection rates  $TR_i^K$  for each row i of the matrix (Table 1). Each rate  $TR_i^K$  is drawn from normal distribution with mean  $TR^K$  and standard deviation  $\beta^K$ . Let F be the interval in which all rejection rates must be drawn. Initially, the lower bound of F is equal to  $TR^K - \beta^K$  and the upper bound to  $TR^K + \beta^K$ . As draws are done, the bounds of F get closer and draws become more restrictive. If the matrix were filled row by row starting from 1 up to N, the last rows would always be drawn in the closest bounds. To cope with this bias, the row number is also drawn randomly. Next, the recognition rates  $TL_i^K$  for each class i (the diagonal elements of  $CM^K$ ) are computed in the same manner but according to the rejection rates already fixed. Each recognition rate  $TL_i^K$  must thus respect two constraints:  $TL_i^K \in [TL_i^K - \alpha^K; TL_i^K + \alpha^K]$  and  $TL_i^K \leq 100 - TR_i^K$ . Finally, the confusion rates  $TC_{i,j}^K$  for each row i and each column j are generated. As for the recognition rates  $TL_i^K$ , the confusion rate  $TC_{i,j}^K$  must be also lower or equal to  $100 - TR_i^K$ .

### 4. Algorithm of output generation

Given the matrix  $CM^K$  as constructed above and the number NS of outputs, the next issue is to generate the list of outputs for each class (Table 2). For that, the matrix of probabilities  $CM^K$  is first transformed into a matrix of numbers  $MN^K$ . This is simply done by multiplying the probabilities in  $CM^K$  for each class  $i$  by the number of solutions  $NS \cdot K$  and then divide them by the sum of rates of this class. Given the matrix  $MN^K$ , we must next assign a list of  $K$  class labels to each output. This could be done completely randomly. But, if the class labels are chosen arbitrarily, the same class label can be drawn several times in the same list of solutions (this would happen for the last outputs to generate). This case is of course impossible to produce by a real classifier. For this reason, we draw the class labels for each output in the most numerous classes.

Now, an output can be composed of one reject label or a list of  $K$  labels. To affect one of these two types of solutions (reject or not) for each output, an integer value  $X$  is randomly drawn in the range  $[1..C_i+R_i]$  where  $C_i$  is the number of remaining class labels and  $R_i$  the number of remaining reject labels for class  $i$  (at the beginning of the output generation,  $C_i$  is initialized to  $\sum_{j=1}^N MN^K[i][j]$  and  $R_i$  to  $MN^K[i][N+1]$ ).  $X \in [1..C_i]$

indicates that a list of  $K$  labels must be generated for this output; otherwise (when  $X \in ]C_i..C_i+R_i]$ ) the output must be rejected.

Actually, when the input value of  $K$  is between 2 and  $N-1$ , a list of at most  $K$  solutions is generated for each output. In other words, the length of the list of solutions is determined for each output. Let  $nlab_i^l$  be the number of class labels to be drawn randomly in the range  $[1..K]$  for the  $l$ th output of class  $i$ . As outputs are treated, it becomes more and more difficult to respect both the constraints imposed by the matrix  $MN^K$  (the choice of labels) and the random draw of  $nlab_i^l$ , particularly for the latest outputs. This can however be avoided by taking into account the number of solutions it remains to generate. This is done using the following principles: let  $S_i^l$  be the  $l$ th output labeled  $i$  among the NS outputs of the class  $i$  ( $i=1..N$ ) and  $O_i^l$  be the number of outputs already treated; the minimum number of possible labels that can be generated for  $S_i^l$  is given by:

$min\_lab = \max(l, C_i - (NS - R_i - O_i^l) \cdot K)$  and  $max\_lab = \min(K, C_i - (NS - R_i - O_i^l))$ . Therefore,  $nlab_i^l$  will be drawn randomly in the range  $[min\_lab..max\_lab]$

The algorithm of the procedure for generating outputs from  $CM^K$  is thus the following (where  $S_{i,j}^l$  is the  $j$ th

solution within the list of solutions associated to output  $S_i^l$  ( $j = 1..nlab_i^l$ ):

### Table 1. Generation of output lists from $CM^K$

**Inputs:**  $MN^K$ : matrix of numbers of labels to generate; NS: number of outputs by class

**Outputs:**  $S_i^l$  for  $i=1$  to  $N$  and  $l=1$  to NS

**Begin**

**For** each class  $i$  from 1 to N **Do**

Initialize  $C_i$  to  $\sum_{j=1}^N MN^K[i][j]$

Initialize  $R_i$  to  $MN^K[i][N+1]$

**While** all outputs are not treated **Do**

Choose a random number  $l$  between 1 and NS

Draw  $X$  randomly in  $[1, C_i+R_i]$

**If**  $X \in ]C_i, C_i + R_i]$  **Then**

Assign the reject solution

Decrement  $R_i$

**Else**

Draw  $nlab_i^l$  in  $[\min\_lab, \max\_lab]$

Choose  $nlab_i^l$  labels from the most numerous classes

Place these labels randomly in  $S_{i,j}^l$

$C_i \leftarrow C_i - nlab_i^l$

**End**

Finally, we associate a random confidence to each generated label within the list of solutions. All the confidences are ordered in a decreasing way. The confidence of the first solution of the outputs must be higher than  $100/nlab_i^l$ . This constraint is available only for outputs which are composed of two solutions at least.

## 5. Experimental results

In this section, we report experiments with a data sets generated by proposed classifier simulator. The main goal of this experiment was to show the importance of simulator in determining the relationship between the performance of combination and the diversity in behaviour of classifiers. In particular, we focus on the performance of sum and mean rule for fusion of ensembles of 3 classifiers exhibiting performance for 10 class problem with no rejection. The repartition of confusion rates in confusion matrix constructed for the generation of classifier outputs is random. For each simulated classifier, 10000 outputs (1000 outputs per class) were generated ( $K=1$ ) and the results was accumulated. Table 2 shows a sample of the results obtained by varying the specified recognition rates on the simulated classifiers; for classifiers all having identical recognition rates ( $\Delta=0$ ) and for classifiers providing different recognition rates ( $\Delta=\{2,$

5,10, 20}). The values of recognition rates combination were averaged over 100 runs.

**Table 2: Combination results with sum rule.**

	Classif. 1	Classif. 2	Classif. 3	$\Delta$	Result
Ens. 1	50	50	50	0	55.54
Ens. 2	50	52	54	2	57.79
Ens. 3	50	55	60	5	61.22
Ens. 4	50	60	70	10	66.83
Ens. 5	50	70	90	20	77.70

The results presented in table 2 show that the recognition rate improvement over the sum rule is achieved only for the sets of classifiers within 5% of each other. It is worth noting that the difference in classifier performance strongly influences the combination results and the sum rule can outperform the best individual classifier for classifiers having near recognition rates. The results presented in table 3 show also the same remark for the simple mean rule which achieves performance improvement only for a balanced ensemble of classifiers (within less than 10% of each other). In particular, the higher is the value of  $\nabla$  (difference between the best and the worst classifiers), the lower is the performance improvement. This idea is confirmed also in [8] on real data for the same combination rule (average).

**Table 3: Combination results with mean rule**

	Classif. 1	Classif. 2	Classif. 3	$\nabla$	Result
Ens. 1	90	88	88	2	93.06
Ens. 2	90	80	81	10	89.39
Ens. 3	90	87	83	8	91.60
Ens. 4	90	75	74	16	86.02
Ens. 5	90	83	74	16	88.35

So, the use of simulator can be considered as an attractive solution to evaluate the combination methods since the simulator can exhibit a performance representative of the result of real classifier.

## 6. Conclusion

In this paper, we have proposed a new method for the artificial generation of classifier outputs based on several parameters which provide sufficient diversity to simulate, for a problem of any number of classes and any type of output, any classifier behaviour. The proposed method relies on a two-step algorithm which first builds a confusion matrix according to a user-specified desired behaviour and secondly generates automatically, from this confusion matrix, outputs of any specified type. We have shown that simulating any classifier behaviour (i.e. any pattern recognition problem) requires the management of the recognition and rejection rates and their standard deviations (to fix the behaviour for each class) but also and at the same time the maximum number of solutions in output lists and the corresponding recognition rates within the lists of solutions.

Thanks to its flexibility and its ability to simulate any classifier behaviour, our simulator will be useful to evaluate all types of combination methods and particularly the combination methods of measurement type which are not sufficiently studied. It is thus an interesting tool that can help to clarify the conditions under which a combination method can be used or is the best for different pattern recognition problems.

## References

- [1] R.P.W. Duin and D.M.J. Tax, "Experiments with classifier combining rules", In *Proc. First Int. Workshop on Multiple Classifier System*, MCS 2000, Vol. 1857, Springer, Berlin, 2000, pp. 16-29.
- [2] T.K. Ho, J.J. Hull and S.N. Srihari, "Decision combination in multiple classifier systems", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.16, No. 1, 1994, pp. 66-75.
- [3] S. Impedovo and A. Salzo, "Evaluation of combination methods", *Proc. ICDAR'99*, Bangalore, India, 1999, pp. 394-397.
- [4] J. Kittler, M. Hatef, R.P.W. Duin and J. Matas, "On combining classifiers", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 3, 1998.
- [5] L.I. Kuncheva and R.K. Kountchev, "Generating classifier outputs of fixed accuracy and diversity", *Pattern Recognition Letters*, Vol. 23, 2002, pp. 593-600.
- [6] V.D. Lecce, G. Dimauro, A. Guerriero, S. Impedovo, G. Pirlo and A. Salzo, "Classifier combination: the role of a-priori knowledge", In *Proc. of the 7<sup>th</sup> International Workshop on Frontiers In Handwriting Recognition*, Amsterdam, The Netherlands, Sept. 11-13, 2000, pp. 143-152.
- [7] J.R. Parker, "Rank and response combination from confusion matrix data", *Information Fusion*, Vol. 2, 2001, pp. 113-120.
- [8] F. Roli, G. Fumera and J.Kittler, "Fixed and trained combiners for fusion of imbalanced pattern classifiers", *The fifth International Conference on Information fusion*, 8-11 July, Annapolis (Washington) USA, 2002.
- [9] D.M.J. Tax, M.V. Breukelen, R.P.W. Duin and J. Kittler, "Combining multiple classifiers by averaging or by multiplying?", *Pattern Recognition*, Vol. 33, 2000, pp. 1475-1485.
- [10] M. Van Erp, L. Vuurpijl and L. Schomaker, "An overview and comparison of voting methods for pattern recognition", *8<sup>th</sup> International Workshop On Frontiers In Handwriting Recognition*, Niagara-on-the-Lake, Ontario, Aug. 6-8, 2002, pp. 195-200.
- [11] B.H. Xiao, C.H. Wang and R.W. Dai, "Adaptive combination of classifiers and its application to handwritten chinese character recognition", In *Proc. ICPR'00*, Barcelona, Spain, Vol. 2, 2000, pp. 2327-2330.
- [12] L. Xu, A. Krzyzak and C.Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition", *IEEE Transaction on Systems, Man, and Cybernetics*, Vol. 22, No 3, 1992, pp. 418-435.